Tampereen ammattikorkeakoulu

Rahansiirtojen valvontaratkaisu ML-analytiikalla Grafanassa

Teemu Vuori

OPINNÄYTETYÖ
Heinäkuu 2025

Dataosaamisen ja tekoälyn ylempi tutkinto-ohjelma

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Dataosaamisen ja tekoälyn ylempi tutkinto-ohjelma
Tradenomi YAMK

VUORI, TEEMU:
Rahansiirtojen valvontaratkaisu ML-analytiikalla Grafanassa

Opinnäytetyö 75 sivua, joista liitteitä 0 sivua
Heinäkuu 2025

Tässä opinnäytetyössä kehitettiin koneoppimista hyödyntävä valvontaratkaisu yrityksen ja pankkien välisten rahansiirtojen seurantaan. Ratkaisu toteutettiin Google Cloud Platformin (GCP) palveluilla, ja sen tavoitteena oli selvittää, kuinka tekoälymenetelmiä voidaan soveltaa reaaliaikaisessa järjestelmävalvonnassa, erityisesti poikkeamien tunnistamiseen ja rahaliikenteen ennustamiseen.

Projektissa rakennettiin tekninen kokonaisuus simuloidun datan pohjalta. Se koostui BigQuery-tietovarastosta ja neljästä analyysimallista (Z-Score-analyysi, Isolation Forest -malli, Vertex AI Forecast -malli ja luokittelumalli). Visualisointiin ja käyttöliittymään käytettiin Grafana Cloud -järjestelmää. Mallien toteutuksessa hyödynnettiin Pythonia kaikissa muissa malleissa paitsi Vertex AI Forecastissa, joka rakennettiin Googlen käyttöliittymän avulla. Mallien tulokset esitettiin Grafanassa erillisissä paneeleissa, ja käyttöliittymä jaettiin yleis-, virhe- ja mallikohtaisiin näkymiin.

Pankkien yksilölliset toimintamallit ja rajallinen näkyvyys niiden järjestelmiin toivat ennustamiseen haasteita. Tästä huolimatta projekti osoitti, että koneoppimismalleilla on potentiaalia tukea valvontaa myös tällaisessa kontekstissa.

Työ toteutettiin aluksi kehittäjän omassa GCP-projektissa ja Grafana Cloud -ympäristössä, mikä mahdollisti joustavamman kehityksen ilman organisaation hallinnollisia rajoitteita. Ratkaisun skaalautuvuus, visuaalisuus ja laajennettavuus tekevät siitä soveltuvan myös muihin järjestelmävalvonnan käyttötapauksiin.

Opinnäytetyön lopputuloksena syntyi toimiva prototyyppi järjestelmä, joka paitsi saavuttu asetetut tavoitteet, luo myös pohjan tekoälypohjaisen valvonnan jatkokehitykselle organisaatiossa.

Asiasanat: koneoppiminen, rahansiirtojen valvonta, google cloud platform, grafana cloud

ABSTRACT

Tampere University of Applied Sciences
Degree Programme in Data Competence and Artificial Intelligence.
Master of Business Administration (MBA)

VUORI TEEMU:
Transaction Monitoring Solution with ML Analytics in Grafana

Master's thesis 75 pages, appendices 0 pages
July 2025

This thesis project developed a machine learning–based monitoring solution for tracking financial transactions between a company and banks. The solution was implemented using services provided by Google Cloud Platform (GCP), with the goal of exploring how artificial intelligence methods can be applied in real-time system monitoring, particularly for anomaly detection and transaction forecasting.

A technical architecture was built based on simulated data. It consisted of a BigQuery data warehouse and four analytical models: Z-Score analysis, an Isolation Forest model, a Vertex AI Forecast model, and a Classification model. Grafana Cloud was used for the user interface and visualization. The models were implemented in Python, except for the Vertex AI Forecast model, which was built using Google's automated service. The results of each model were displayed in Grafana using dedicated panels, and the user interface was organized into general, error-specific, and model-specific views.

Individual operational patterns of banks and limited visibility into their internal systems posed challenges for forecasting. Nevertheless, the project demonstrated that machine learning models hold significant potential in supporting monitoring even in such constrained contexts.

The development was initially carried out in the developer's personal GCP project and Grafana Cloud environment, allowing for more flexible development without administrative restrictions from the organization. The solution's scalability, visual clarity, and extensibility make it suitable for other system monitoring use cases as well.

As a result, the thesis produced a working prototype system that not only achieved the defined objectives but also lays the groundwork for future development of AI-driven monitoring within the organization.

Key words: machine learning, transaction monitoring, google cloud platform, grafana cloud

SISÄLLYS

ERITYISSANASTO

API	Application Programming Interface - sovellusrajapinta
ARIMA	AutoRegressive Integrated Moving Average – aikasarjaennustemalli
AutoML	Automated Machine Learning – automaattinen koneoppiminen
BI	Business Intelligence – liiketoimintatiedon hallinta
BigQuery	GCP:n tietovarastopalvelu
CD	Continuous Delivery – jatkuva toimitus
CI	Continous Integration – jatkuva integrointi
Cloud Run	GCP:n konttiajoalusta – palvelu konttien suorittamiseen ilman palvelinten hallintaa
CSV	Comma-Separated Values – pilkuin eroteltu tietostomuoto
Dataflow	GCP:n datankäsittelytyökalu
DBSCAN	Density-Based Spatial Clustering of Aplication with Noice – tiheyspohjainen klusterointialgoritmi
DVS	Data Visualization Society – tiedon visualisoinnin yhteisö
GMP	Google Marketing Platform – Googlen markkinointialusta
GUI	Graphical User Interface – graafinen käyttöliittymä
IAM	Identity and Access Management - Käyttöoikeuksien hallintajärjestelmä
JSON	JavaScript Object Notation – Kevyt tiedonsiirron muoto

KPI	Key Performance Indicators – suorituskykymittari
Notebook	Interaktiivinen koodausympäristö (esim Jupyter tai Vertex AI Workbench)
OLAP	Online Analytical Processing – verkkopohjainen analytiikkaprosessointi
PR-AUC	Precision-Recall Area Under Curve – luokittelumallin tarkkuus-herkkyys-mittari
Prometheus	Monitorointijärjestelmä – aikasarjatietokanta ja keräystyökalu
Pub/Sub	Publish/Subscibe - viestinvälitysjärjestelmä (GCP)
ROC-AUC	Receiver Operating Characteristic – Area Under Curve – luokittelumallin tarkkuus-herkkyys-mittari
SQL	Structured Query Language – tietokantojen kyselykieli
SVM	Support Vector Machine - luokittelualgoritmi
Vertex AI	Google Cloudin koneoppimisalusta – hallittu ML-kehitysympäristö
Z-score	Z-arvo – poikkeaman mittari standardipoikkeamana

1 JOHDANTO

1.1 Projektin konteksti ja liiketoimintalähtöinen motiivi

Tämä opinnäytetyö lähti liikkeelle tarpeesta kehittää reaaliaikainen ja ennakoiva valvontaratkaisu pankkien ja yrityksen välisten rahansiirtojen seuraamiseen. Lähtökohtana oli yrityksessä havaittu kehityskohde: perinteiset valvontaratkaisut eivät tarjonneet riittävää kykyä havaita poikkeamia ajoissa tai tukea päätöksentekoa dynaamisessa toimintaympäristössä. Haluttiin rakentaa järjestelmä, joka ennustaa poikkeustilanteita ennakkoon analysoimalla tapahtumia, mutta jossa varsinainen reagointi ja toimenpiteet jäävät edelleen ihmisten tehtäväksi. Näin yhdistetään automaattisen analyysin nopeus ja ihmisen harkinta tilanteiden käsittelyssä.

Koneoppimisen käyttöönottoa valvontatehtävissä ei ollut aiemmin testattu yrityksessä, mutta sen mahdollisuudet tunnistettiin osaksi pidemmän aikavälin digitalisaatiostrategiaa. Tämä tarjosi luontevan tilaisuuden yhdistää henkilökohtainen kiinnostukseni koneoppimiseen ja tekoälyn hyödyntämiseen liiketoimintakriittisissä prosesseissa käytännönläheiseen kehitysprojektiin. Työssä pyrittiin myös vahvistamaan omaa teknistä osaamista erityisesti Google Cloud Platformin palveluissa ja Grafana Cloudin käytössä.

Päätös rakentaa ratkaisu juuri GCP:hen pohjautuen perustui kolmeen keskeiseen tekijään:

1. GCP:n tarjoamat hallitut koneoppimispalvelut mahdollistivat nopean prototyyppien rakentamisen ilman laajaa infrastruktuurityötä,
2. Organisaatiossa aikaisempi kokemus GCP:n muista palveluista madalsi käyttöönoton kynnystä.
3. Kustannustehokkaat tai osittain ilmaiset työkalut, kuten BigQuery ja Vertex AI:n ennustemalli, sopivat hyvin opinnäytetyön resurssiraameihin. Muita alustoja, kuten AWS tai Azure, ei valittu, koska niissä vastaavien työkalujen käyttöönotto olisi vaatinut enemmän konfigurointia tai niistä olisi aiheutunut korkeampia kustannuksia prototyyppivaiheessa.

1.2 Tavoitteet, rajaukset ja käytännön hyödyt

Tavoitteena oli rakentaa end-to-end-prototyyppi, joka osoittaa, kuinka koneoppimista voidaan hyödyntää valvontatehtävissä konkreettisesti. Tavoite ei ollut kehittää tuotantovalmista järjestelmää vaan tutkia, miten eri teknologiat voidaan yhdistää toimivaksi kokonaisuudeksi ja millaisia etuja koneoppiminen tarjoaa poikkeamien havaitsemisessa ja rahavirtojen seuraamisessa. Rajauksena oli käyttää pelkästään (todenmukaista) simuloitua dataa, jotta vältettäisiin henkilötietoihin tai liiketoiminnallisesti arkaluonteisiin tietoihin liittyvät ongelmat, vaikka se ei ihan täyttä varmuutta mallien toiminnasta antaisikaan.

Prototyypin keskeinen hyöty on sen käyttöliittymässä: Grafanaan rakennettu dashboard mahdollistaa reaaliaikaisen näkymän rahansiirtoihin eri näkökulmista. Käyttäjät voivat seurata pankkikohtaisia tapahtumia, poikkeamien määrää ja ennusteiden osumatarkkuutta. Integroimalla mallien tulokset samaan näkymään saadaan kattava yleiskuva, joka tukee sekä nopeaa reagointia että syvällisempää analyysiä.

1.3 Tutkimus- ja kehityskysymysten määrittely

Opinnäytetyön tarkoituksena oli selvittää, voidaanko pilvipohjaisilla koneoppimisratkaisuilla tukea organisaation järjestelmävalvontaa tavalla, joka on skaalautuva, automatisoitu ja visuaalisesti selkeä. Tavoitteena oli myös saada käsitys siitä, missä tilanteissa eri mallit toimivat parhaiten ja miten niiden yhdistäminen parantaa kokonaisvalvontaa.

Koska ennustaminen perustuu simuloituun dataan eikä pankkien sisäisiin algoritmeihin ole pääsyä, oli tärkeää arvioida realistisesti mallien kyvykkyyttä toimia puhtaasti ulkoisesti havaittavan tiedon varassa. Tämä edellytti kriittistä tarkastelua sekä mallien valinnassa että arviointikriteereissä. Kysymyksiä, joihin etsittiin vastauksia:

- Miten pankkien ja yrityksen välisiä rahansiirtoja voidaan seurata reaaliaikaisesti ja visuaalisesti?
- Voiko koneoppimismalleja hyödyntää poikkeamien havaitsemisessa ja tulevien rahavirtojen ennustamisessa?
- Miten Grafana Cloud, BigQuery ja Vertex AI voidaan yhdistää kustannustehokkaaksi ja käytännössä toimivaksi koneoppimiseen perustuvaksi valvontaratkaisuksi?

1.4 Menetelmät ja datan käsittely

Tutkimusmenetelmänä hyödynnettiin kehittämistutkimusta, jossa tekninen ratkaisu rakennettiin vaiheittain konkreettisen käyttötapauksen ympärille. Menetelmällisesti työssä korostui käytännönläheinen lähestymistapa: pyrittiin rakentamaan toimiva kokonaisuus, jossa data, mallit ja visualisointi kytkeytyvät saumattomasti toisiinsa.

Data luotiin itse Pythonilla simuloiden, mikä tarjosi täyden hallinnan tiedon rakenteeseen ja mahdollisti sellaisten ilmiöiden, kuten viiveiden tai poikkeamien, tietoinen sisällyttäminen mallien testaukseen. BigQuery toimi tietovarastona, josta kaikki mallit lukivat tietonsa. Datan esikäsittely ja aggregointi toteutettiin pääosin SQL-kyselyillä, mutta monimutkaisemmat datanmuokkaukset tehtiin Pythonilla VS Codella tai Jupyter-notebookeissa Vertex AI Workbenchin kautta. Tämä mahdollisti tehokkaan ja koodipohjaisen lähestymistavan mallien rakentamiseen.

Koneoppimismallit (Z-Score, Isolation Forest, luokittelumalli) toteutettiin koodilla Pythonin avulla, mikä mahdollisti tarkemman kontrollin malliparametreihin ja tulosten käsittelyyn. Poikkeuksena oli Vertex AI Forecast -malli, joka rakennettiin Googlen käyttöliittymäpohjaisella palvelulla ilman omaa koodia. Näin saatiin mukaan myös ns. low-code/no-code -lähestymistapa, jota voidaan hyödyntää esimerkiksi tilanteissa, joissa ohjelmointiosaamista ei ole käytettävissä. Visualisointiin käytettiin Grafanaa, joka yhdistettiin BigQueryyn SQL-kyselyiden avulla.

Lopputuloksena syntyi järjestelmä, jossa sekä automatisoitu analyysi että visuaalinen valvonta tukevat toisiaan ja jossa mallien ylläpito ja jatkokehitys on helposti toteutettavissa ilman raskasta manuaalista työtä.

2 TEOREETTINEN VIITEKEHYS

2.1 ML-mallityypit: valvottu, valvomaton ja aikasarjamallit

Opinnäytetyön tavoitteena oli rakentaa koneoppimista hyödyntävä valvontaratkaisu, joka tunnistaa poikkeamia, tekee ennusteita ja luokittelee rahansiirtoihin liittyviä tapahtumia. Tätä varten tarkasteltiin kolmea koneoppimisen päätyyppiä: valvottua oppimista, valvomatonta oppimista sekä aikasarja-analytiikkaa. Jokaisella lähestymistavalla on omat vahvuutensa eri tyyppisten ongelmien ratkaisemisessa. Valvottu oppiminen perustuu opetusdatan mukana tuleviin valmiisiin vastauksiin, joita käytetään mallin kouluttamiseen tyypillisesti luokitteluun tai regressioon. Valvomaton oppiminen toimii ilman opetusdataan liitettyjä vastauksia, ja sen avulla pyritään tunnistamaan esimerkiksi poikkeamia tai datassa esiintyviä epätavallisia rakenteita. Aikasarjamallit taas soveltuvat parhaiten ennustamiseen ajassa etenevässä datassa. (Géron, 2023 s. 41–64, 163–184, 529–558.)

Tässä projektissa päädyttiin hyödyntämään kaikkia kolmea lähestymistapaa kattavan näkymän saavuttamiseksi. Jokainen malli valittiin huolellisesti vertaillen sen soveltuvuutta rahansiirtojen valvontaan, analysoiden myös vaihtoehtoisia ratkaisuja. Esimerkiksi valvomattomaan oppimiseen soveltuvia vaihtoehtoja olisivat olleet muun muassa DBSCAN tai One-Class SVM, mutta ne eivät tarjonneet yhtä hyvää skaalausta suurille tietomäärille kuin Isolation Forest. Ennustamiseen olisi voitu käyttää myös Metan (Facebook) kehittämää Prophet-mallia tai ARIMA-malleja, mutta Vertex AI Forecast tarjosi valmisintegraation GCP-ympäristöön, automaattisen hyperparametrisäädön ja mahdollisuuden hyödyntää Googlen automaattista mallinrakennusta. Tämä mahdollisti valmiin mallin luomisen ilman syvällistä parametrien manuaalista virittämistä ja tarjosi samalla mahdollisuuden arvioida, kuinka hyvin Googlen automatisoitu ratkaisu toimii käytännön seurannassa. Valvottuun luokitteluun olisi ollut vaihtoehtoja, kuten Vertex AI:n tarjoama AutoML-palvelu, joka automatisoi mallinrakennuksen ja voi hyödyntää esimerkiksi XGBoost-algoritmia. Tässä työssä haluttiin kuitenkin tarkempi hallinta ja mahdollisuus manuaaliseen säätöön Pythonilla toteutetun mallin avulla.

2.2 Poikkeavuuksien tunnistaminen: Z-Score ja Isolation Forest

Poikkeavuuksien tunnistaminen on keskeinen osa järjestelmävalvontaa, erityisesti rahansiirtojen kaltaisissa ajassa tapahtuvissa prosesseissa. Tässä työssä hyödynnettiin kahta erilaista lähestymistapaa poikkeamien tunnistamiseen: sääntöpohjainen Z-Score analyysi sekä koneoppimiseen perustuva Isolation Forest malli.

Z-Score on yksinkertainen mutta tehokas tilastollinen menetelmä, jossa mitataan yksittäisten tapahtumien poikkeamaa historiallisesta keskiarvosta. Mallin etuna on sen nopea laskettavuus ja selkeä tulkittavuus, mikä tekee siitä hyvän vertailutason eli baseline mallin (Hyndman & Athanasopoulos, 2021, s. 27.) Grafana käyttöliittymässä Z Score mahdollistaa nopean visuaalisen reagoinnin. Käyttäjä näkee selkeästi, kun jokin arvo ylittää normaalin vaihteluvälin. Tämä helpottaa valvojia kohdistamaan huomionsa niihin ajankohtiin ja pankkeihin, joissa rahavirrat ovat merkittävästi poikenneet odotetusta. Se toimii erityisesti tilanteissa, joissa tarvitaan yksinkertainen mutta tehokas mittari äkillisten muutosten havaitsemiseksi.

Isolation Forest malli puolestaan perustuu valvomattomaan oppimiseen ja käyttää päätöspuista muodostettuja rakenteita poikkeamien eristämiseen. Sen vahvuus on kyky tunnistaa monimutkaisempia poikkeamia, joita Z-Score ei tavoita, kuten monimuuttujaisia äkillisiä muutoksia (Aggarwal, 2017, s. 339-342.) Grafanassa malli tuo mukanaan dynaamisempaa valvontaa. Se oppii automaattisesti normaalin toimintaympäristön ja reagoi sen muuttuessa. Tämä vähentää tarvetta manuaalisten raja-arvojen määrittämiselle ja tekee järjestelmästä mukautuvamman nopeasti muuttuvissa tilanteissa, kuten kausivaihteluissa tai teknisissä häiriöissä. Isolation Forest on hyödyllinen silloin, kun ei ole olemassa selkeää sääntöpohjaista tapaa määrittää, milloin jokin tilanne on poikkeava.

2.3 Aikasarja-analytiikan perusteet: ennustamisen lähtökohdat

Rahavirtojen seuranta on luonteeltaan ajassa etenevää ja kausivaihteluille altista toimintaa. Siksi aikasarja-analytiikka nousi keskeiseen rooliin. Aikasarjojen analyysi tarjoaa keinon tarkastella rahansiirtojen käyttäytymistä eri aikajaksoilla ja mahdollistaa ennakoivan näkymän rakentamisen. Ennustamisen perustana ovat aikaisemmat tapahtumat, joiden pohjalta voidaan arvioida tulevaa kehitystä.

Grafanassa aikasarjapohjainen lähestymistapa mahdollistaa siirtomäärien ja summien visualisoinnin eri aikaväleillä. Tämä auttaa valvojia tunnistamaan sekä tavanomaiset että poikkeavat kehityssuunnat. Lisäksi aikasarja-analyysi mahdollistaa aggregoidun datan tarkastelun pankki- ja aikakohtaisesti, mikä antaa operatiivisille tiimeille nopean yleiskuvan tilanteesta. Aikasarja-analytiikka luo siis perustan jatkuvalle tarkkailulle ja toimii pohjana ennakoiville malleille. Se helpottaa havaintojen tekemistä ilman että yksittäisiä lukuja tarvitsee tulkita manuaalisesti (*Grafana Labs, 2025c*)

2.4 Aikasarjapohjainen ennustaminen: Vertex AI Forecast -malli

Vertex AI Forecast on Googlen tarjoama aikasarjaennustemalli, joka tarjoaa automaattisen mallinrakennuksen, hyperparametrisäädön sekä valmiin integraation GCP-ympäristöön. (*Google Cloud, 2025b*) Malli valittiin ennustavan analytiikan välineeksi. Malli koulutettiin simuloidulla historiallisella rahansiirtodatalla, ja sen tavoitteena oli arvioida tulevien tuntien tai päivien rahavirtojen suuruus. Ennustamisen avulla voidaan havaita tilanteet, joissa siirtomäärät tai summat poikkeavat odotetusta kehityksestä.

Grafanassa ennustemallin tuottamat tulokset esitettiin yhdessä toteutuneen historian kanssa, jolloin käyttäjä pystyy yhdellä silmäyksellä arvioimaan, ylittävätkö tai alittavatko nykyiset siirtomäärät ennustetun tason. Tämä mahdollistaa varhaisen reagoinnin mahdollisiin ongelmiin esimerkiksi kun odotettu siirtomäärä ei toteudukaan. Vertex AI Forecastin etuna on lisäksi sen

automaattinen skaalautuvuus ja hyperparametrisäätö, jolloin mallin ylläpito ei vaadi jatkuvaa manuaalista säätämistä. Lisäksi valmiin integraation ansiosta mallin tuotokset saatiin helposti liitettyä BigQueryn kautta Grafanaan. Tämä tekee ennusteiden seuraamisesta sujuvaa ja läpinäkyvää ilman erillistä teknistä ylläpitoa

2.5 Valvottu luokittelumalli: Classification Pythonilla

Viimeisenä mallina toteutettiin manuaalisesti rakennettu luokittelumalli Pythonilla. Kyseessä on valvottuun oppimiseen perustuva malli, joka opetettiin luokittelemaan tapahtumia esimerkiksi normaaleiksi tai poikkeaviksi. Malli hyödynsi opetusaineistoa, jossa tapahtumat oli etukäteen luokiteltu. Se käytti useita muuttujia, kuten siirtosummaa, pankkia, ajankohtaa ja siirtotiheyttä oppiakseen, millaiset tapahtumat ennakoivat poikkeavuuksia.

Grafanassa mallin hyöty tulee esiin erityisesti tapahtumien jälkikäteisarvioinnissa ja luokittelussa. Paneelien kautta voidaan nopeasti tarkastella, kuinka suuri osa tapahtumista luokiteltiin poikkeaviksi ja millä luottamustasolla. Tämä tukee päätöksentekoa silloin, kun operatiivisen tiimin on tehtävä valintoja reagoinnin tai lisäselvityksen tarpeesta. Python-malli antoi myös mahdollisuuden säätää mallin rakennetta tarkasti käsin, mikä oli tärkeää, kun haluttiin ymmärtää tarkemmin mallin toimintaperiaatetta ja tulkittavuutta (Géron, 2023, s. 530-531). Malli on käytännössä hyödyksi tilanteissa, joissa poikkeamien syitä halutaan ymmärtää tarkemmin ja tehdä läpinäkyvä luokittelu epäilyttäville tapahtumille.

Kaikkien neljän mallin yhdistäminen samaan Grafana pohjaiseen käyttöliittymään mahdollisti kokonaisvaltaisen ja monikerroksisen valvontaratkaisun rakentamisen, joka tukee sekä reaaliaikaista että ennakoivaa analytiikkaa eri näkökulmista.

2.6 Aiemmat vastaavat ratkaisut ja tutkimukset

Koneoppimisen hyödyntäminen pankkitapahtumien ja erityisesti reaaliaikaisten rahansiirtojen valvontaan on yleistynyt viime vuosina sekä tutkimuksessa että käytännön sovelluksissa. Kansainvälisessä tutkimuksessa on tarkasteltu erilaisia koneoppimismalleja, joilla pyritään tunnistamaan poikkeamia suurista transaktiomassoista, ja erityisesti valvomattomat algoritmit, kuten Isolation Forest, ovat osoittautuneet lupaaviksi.

Vaikka täysin samanlaista toteutusta kuin tämän opinnäytetyön, jossa yhdistyvät reaaliaikainen rahansiirtovalvonta, koneoppimismallit, Google Cloud Platform ja Grafana – ei ole suomalaisessa ammattikorkeakoulukentässä aiemmin dokumentoitu, on olemassa tutkimuksia ja opinnäytetöitä, jotka sivuavat aihetta ja tukevat työn taustaa eri näkökulmista.

Dehnavin opinnäytetyössä tarkasteltiin koneoppimismallien käyttöä pankkitapahtumien poikkeamien tunnistamisessa. Vaikka lähestymistapa ja käytetty tekninen ympäristö eroavat tämän työn ratkaisusta, työ osoittaa kiinnostuksen kasvavan myös suomalaisissa ammattikorkeakouluissa rahoitusdatan analysointiin koneoppimisen keinoin (Dehnavi, 2025)

Kulmalan opinnäytetyö käsitteli koneoppimisen ja tietokonenäön hyödyntämistä kodin kameravalvonnassa. Vaikka konteksti eroaa rahoitusalasta, siinä hyödynnettiin reaaliaikaisia älymalleja, mikä tekee siitä relevantin esimerkin valvonnan modernista kehityksestä (Kulmala, 2024.) Samoin Anttilan työ, joka käsittelee raideliikenteen riskiperusteista valvontaa Excel-pohjaisella työkalulla, toimii esimerkkinä dataan perustuvan operatiivisen valvonnan kehittämisestä. (Anttila, 2024)

Kansainvälisesti tarkasteltuna Lässig on julkaissut käytännönläheisen oppaan A Guide to Building a Financial Transaction Anomaly Detector, jossa esitellään valvomattomien mallien, kuten Isolation Forestin, hyödyntämistä finanssidatan valvonnassa. Tämä julkaisu tarjoaa mallikehyksen reaaliaikaisille toteutuksille, joita tämäkin työ osaltaan pyrkii soveltamaan (Lässig,2021.) Lisäksi Soria, Loayza Abal ja Segura Peña kokosivat systemaattisen kirjallisuuskatsauksen

ML-mallien soveltamisesta rahanpesun tunnistamiseen, käsitellen esimerkiksi SVM-, KNN-, päätöspuu- ja syväoppimismenetelmiä (Soria, Loayza Abal, & Segura Peña, 2024.)

Yhteenvetona voidaan todeta, että vaikka täysin vastaavaa toteutusta ei ole aiemmin dokumentoitu, olemassa oleva kotimainen ja kansainvälinen tutkimus osoittaa selvästi, että koneoppimisen ja valvontajärjestelmien yhdistäminen rahoitusalan kontekstissa on ajankohtainen ja merkityksellinen tutkimusalue. Tämä opinnäytetyö täydentää olemassa olevaa kenttää tuomalla uuden näkökulman Grafana- ja GCP-pohjaiseen valvontaratkaisuun.

3 JÄRJESTELMÄSUUNNITTELU JA ARKKITEHTUURI

3.1 Käytetyt teknologiat ja kehitysympäristö

Toteutettavan järjestelmän teknologinen kokonaisuus suunniteltiin alusta alkaen modulaariseksi ja pilvipohjaiseksi, jotta eri osa-alueet, kuten datan käsittely, analytiikka ja visualisointi, voitiin toteuttaa toisistaan riippumattomasti, mutta silti yhtenäisesti integroituvina kokonaisuuksina. Tämän lähestymistavan etuna oli erityisesti kehitystyön joustavuus, mahdollisuus komponenttien erilliseen päivittämiseen sekä koko järjestelmän skaalautuvuus tarpeen mukaan.

Keskeiseksi alustaksi valittiin Google Cloud Platform (GCP), joka tarjoaa laajan valikoiman palveluita erityisesti datan käsittelyyn ja analytiikkaan. GCP:n valintaan vaikuttivat sekä sen tekninen kypsyys että käytettävissä olevien työkalujen integraatiomahdollisuudet. Ympäristön ytimen muodostivat BigQuery ja Vertex AI -palvelut. BigQuery toimi projektissa ensisijaisena tietovarastona, joka mahdollisti suurten tapahtumamäärien tehokkaan analysoinnin suoraan SQL-kyselyillä. Koska järjestelmässä haluttiin mahdollistaa lähes reaaliaikainen seuranta, BigQueryn suorituskyky ja kyky vastata ajallisesti kriittisiin kyselyihin nousivat keskeiseen rooliin.

Vertex AI Workbench tarjosi puolestaan hallitun Jupyter-pohjaisen kehitysympäristön, jossa koneoppimiseen liittyvät toiminnot, kuten mallien opetus, testaus ja arviointi, voitiin suorittaa keskitetysti ja turvallisesti. Workbenchin etuna oli myös sen valmius hyödyntää suoraan muita GCP:n palveluita ilman monimutkaista autentikointia tai siirtomekanismeja, mikä yksinkertaisti kehitystyötä merkittävästi. Kehitystyössä hyödynnettiin Python-ohjelmointikieltä, jonka laaja kirjastoekosysteemi (esimerkiksi pandas, scikit-learn, matplotlib ja Google Cloudin viralliset asiakaskirjastot) tarjosi kaiken tarvittavan mallien rakentamiseen ja analyysien suorittamiseen (Raghavendra, 2023, s. 158-175.)

Paikallista koodinkehitystä ja konfiguraatioiden muokkaamista varten käytössä oli Visual Studio Code -editori, joka tarjosi kevyen mutta tehokkaan

käyttöliittymän Python-kehitykselle. Editorin joustava laajennettavuus, versiohallintaintegraatio ja hyvä tuki muun muassa Jupyter-notebookeille tekivät siitä sopivan työkalun tilanteisiin, joissa haluttiin työskennellä nopeasti ja ilman täyttä pilviresurssien latautumista.

Visualisointikerros toteutettiin Grafana Cloud -ympäristössä, joka valittiin sen käyttövalmiin selainpohjaisuuden, BigQuery-yhteensopivuuden ja aiemman käytön vuoksi. Grafana Cloudin käyttö mahdollisti koko järjestelmän pääkäyttöliittymän toteuttamisen ilman tarvetta ylläpitää omaa palvelininfrastruktuuria. Paneeleiden dynaaminen päivittyminen, muuttujien hyödyntäminen sekä selkeä rakenne tukivat juuri sellaista visuaalista kokonaisuutta, joka oli tavoitteena: yksinkertainen, nopea ja käytännöllinen näkymä tapahtumiin ja mallien tuloksiin.

Kaikkien järjestelmään liittyvien koodien, konfiguraatioiden ja visualisointien hallintaan käytettiin GitHubia. Sen avulla varmistettiin versioiden hallinta, muutosten jäljitettävyys sekä mahdollisuus kehittää järjestelmää usean eri näkökulman ja kehittäjän voimin. Tämä rakenne osoittautui erityisen hyödylliseksi, koska järjestelmää kehitettiin sekä pilvessä että paikallisesti, ja yhteisen lähdekoodin hallinta oli välttämätöntä jatkuvuuden kannalta.

3.1.1 Google Cloud Platform (GCP)

Google Cloud Platform (GCP) valittiin projektin pilvipalvelualustaksi sen tarjoaman suorituskyvyn, skaalautuvuuden ja integroitavuuden vuoksi. GCP:n palveluarkkitehtuuri mahdollisti modernin ja modulaarisen järjestelmän rakentamisen, jossa keskeiset komponentit, kuten tietovarastointi, laskentaresurssit ja käyttöliittymäintegraatiot, toimivat yhtenäisessä pilviympäristössä.

Datan tallennukseen ja analysointiin käytettiin GCP:n tarjoamaa BigQuery-tietovarastoa, joka mahdollisti suuren tapahtumamäärän tehokkaan hallinnan ja käsittelyn SQL-pohjaisten kyselyiden avulla. BigQueryn etuna oli sen kyky käsitellä massadataa nopeasti ja skaalautuvasti ilman omaa

palvelininfrastruktuuria. Lisäksi sen tiivis integraatio muihin GCP-palveluihin yksinkertaisti tiedonsiirtoa ja mahdollisti rajapintapohjaisen arkkitehtuurin rakentamisen (*Google*, 2025a.)

Vaikka opinnäytetyön toteutuksessa ei hyödynnetty GCP:n käyttöoikeushallintaa laajasti, oli merkittävää, että GCP tarjoaa sisäänrakennetun Identity and Access Management (IAM) -järjestelmän. Tämä mahdollistaa roolipohjaisen pääsynhallinnan eri resursseihin ja varmistaa tietoturvallisen kehitysympäristön yrityksen tulevassa tuotantokäytössä. IAM:n olemassaolo tarjoaa valmiuden hallita käyttöoikeuksia tarkasti, mikä tukee myös auditointivaatimuksia ja tietosuojakäytäntöjä.

Järjestelmä suunniteltiin siten, että se olisi helposti laajennettavissa yritysympäristön tuleviin tarpeisiin. GCP:n laaja palvelutarjonta, kuten Cloud Run taustaprosessien ajamiseen, Pub/Sub viestinvälitykseen sekä Dataflow automatisoitujen dataputkien hallintaan, mahdollistavat järjestelmän jatkokehittämisen ilman merkittäviä arkkitehtuurimuutoksia. Näiden palveluiden avulla on mahdollista rakentaa täysin automatisoitu ja reaaliaikainen dataputki esimerkiksi virheilmoitusten tai tapahtumapohjaisen malliajon tueksi (*Google*, 2025b).

Kaiken kaikkiaan GCP tarjosi teknisesti vakaan ja skaalautuvan alustan järjestelmän toteutukselle, mahdollistaen tehokkaan kehityksen, tietoturvallisen hallinnan ja selkeän etenemispolun kohti tuotantovalmiutta.

3.1.2 Koneoppimismallien suunnittelu ja kehitysympäristö

Koneoppimismallien kehitystyö suunniteltiin toteutettavaksi pääosin Python-ohjelmointikielellä, joka tarjosi laajan ja kypsän ekosysteemin tilastolliseen analyysiin, mallinnukseen ja visualisointiin. Pythonin käyttö mahdollisti joustavan ja läpinäkyvän lähestymistavan mallien suunnitteluun ja kehittämiseen, mikä oli tärkeää erityisesti tämän opinnäytetyön käytännönläheisessä mutta tutkimuksellisia elementtejä sisältäneessä kontekstissa. Manuaalinen mallien rakentaminen koettiin hyödylliseksi, sillä se auttoi ymmärtämään syvällisemmin

datan käyttäytymistä, valittujen menetelmien taustalla olevaa matematiikkaa ja mallin toiminnallista logiikkaa. Tämä lähestymistapa myös mahdollisti yksityiskohtaisemman kontrollin mallin rakenteeseen, hyperparametreihin ja virheenkäsittelyyn.

Kehitystyössä hyödynnettiin kahta rinnakkaista työympäristöä: paikallista Visual Studio Code -editoria sekä Google Cloud Platformin tarjoamaa Vertex AI Workbench -notebook-ympäristöä. Visual Studio Code toimi tehokkaana työkaluna paikalliseen koodinkirjoittamiseen ja versionhallintaan, kun taas Vertex AI Workbench tarjosi hallitun pilvipohjaisen kehitysympäristön, jonka kautta oli mahdollista suorittaa laskennallisesti raskaampia prosesseja suoraan GCP:n infrastruktuurissa. Workbenchin valmiit koneoppimiskirjastot sekä tiivis integraatio BigQueryn kanssa nopeuttivat merkittävästi kokeellista kehitystä ja tulosten palauttamista takaisin analytiikkaan.

Mallien suunnittelussa kiinnitettiin huomiota myös niiden käyttöön tulevassa valvontajärjestelmässä. Kehitystyön alkuvaiheessa määriteltiin, mitkä mallityypit soveltuvat parhaiten järjestelmän tavoitteisiin – kuten poikkeamien tunnistamiseen, ennustamiseen ja luokitteluun, sekä miten ne voidaan yhdistää BigQueryn ja Grafanan kanssa saumattomaksi kokonaisuudeksi. Tämä edellytti myös arkkitehtuurin yhteensopivuuden huomioimista: esimerkiksi mallien tulokset tuli pystyä palauttamaan takaisin BigQuery-tauluihin ja visualisoimaan ne käyttöliittymässä ilman manuaalisia välikäsittelyitä.

On myös tärkeää huomioida, että opinnäytetyön yhteydessä käytössä ei ollut tuotantodataa, vaan kaikki koneoppimismallit suunniteltiin ja testattiin simuloidun datan pohjalta. Simuloinnilla pyrittiin luomaan mahdollisimman realistinen aineisto, joka vastaa tuotantoympäristön rakenteita ja sisältöä. Tämä mahdollisti sen, että kehitetyt mallit voidaan myöhemmin siirtää suoraan tuotantokäyttöön ilman merkittäviä rakenteellisia muutoksia, kunhan niille tarjotaan vastaavan muotoista todellista dataa.

Lopputuloksena syntyi järjestelmä, jossa koneoppimismallit voitiin ajaa joko erillisinä Python-skripteinä tai hyödyntäen pilvipalveluiden valmiita työkaluja, kuten Vertex AI Forecast -palvelua. Valittu yhdistelmämalli tarjosi sekä

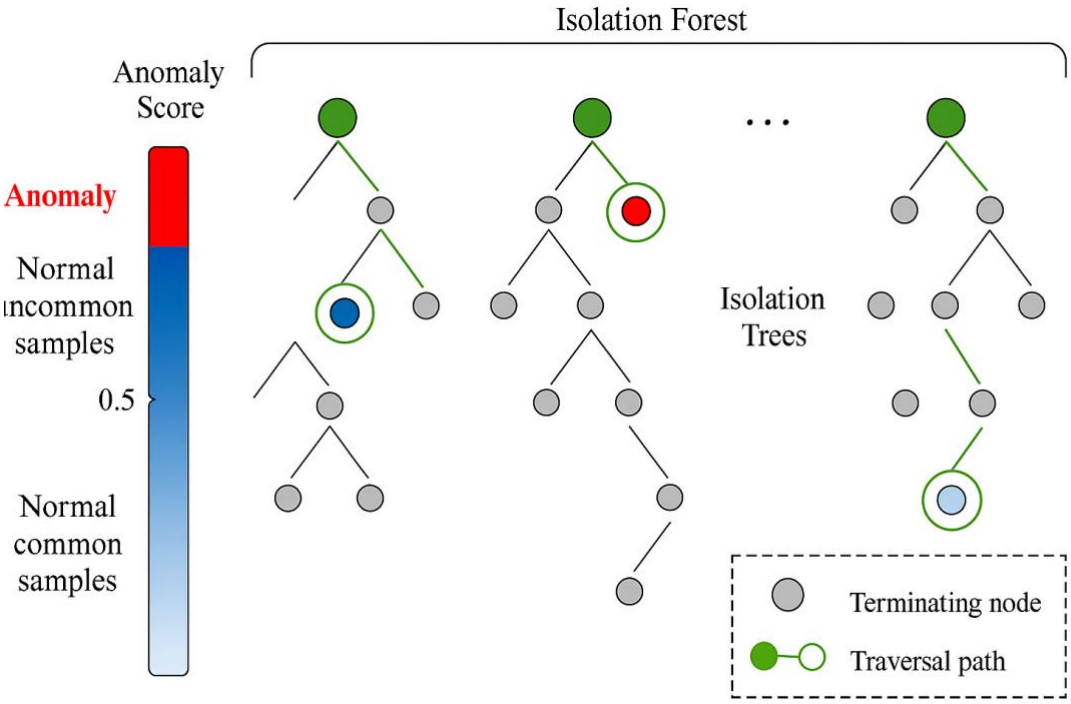
joustavuutta kehitystyöhön että skaalautuvuutta tulevaisuuden tuotantokäyttöä varten.

3.1.3 Python-mallit

Pythonilla toteutettujen koneoppimismallien kehitysprosessi perustui yhtenäiseen ja toistettavaan datapohjaan, joka oli suunniteltu siten, että kaikki mallit voisivat hyödyntää samoja simuloituja syötteitä ilman suuria rakenteellisia muutoksia. Simuloitu data ladattiin suoraan BigQuerystä `google-cloud-bigquery`-kirjaston avulla, minkä jälkeen se käsiteltiin `pandas`-kirjastolla edelleen mallinnukseen sopivaan muotoon. Esikäsittelyyn kuului puuttuvien arvojen tarkistus, aikaleimojen standardointi sekä tarvittavien muuttujien luokittelu ja muunnos numeeriseen muotoon (McKinney, 2022, s. 28-30.)

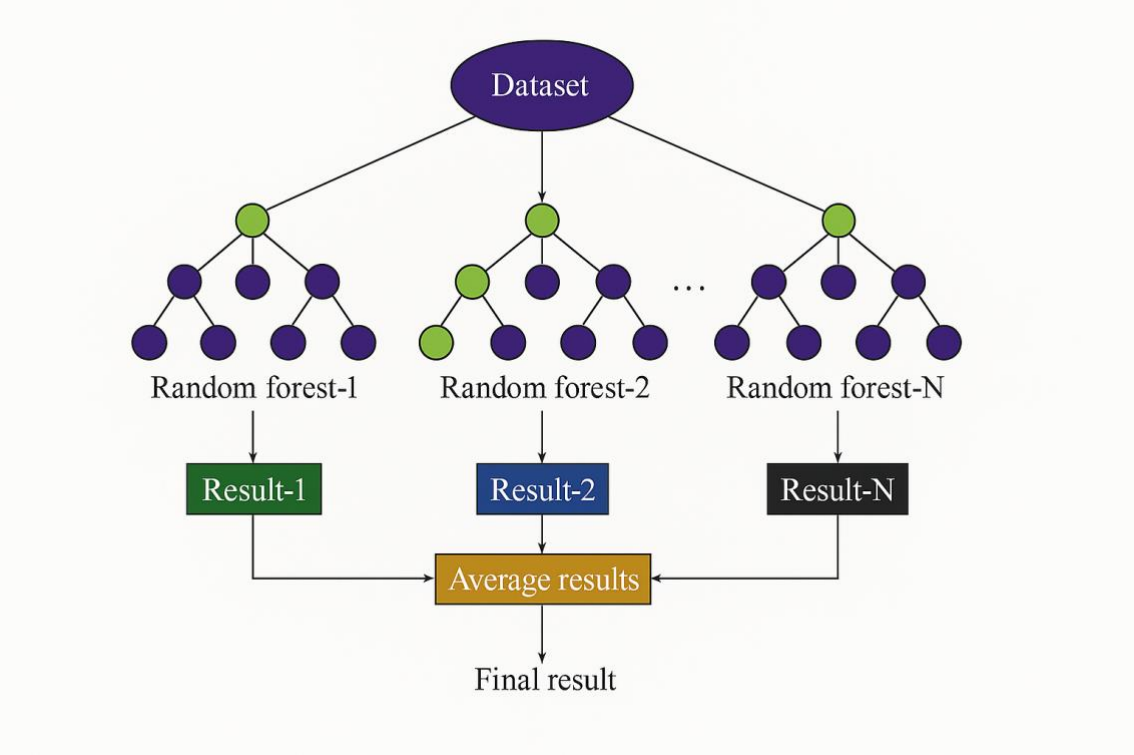
Ensiksi toteutettiin yksinkertainen tilastollinen Z-score-malli, joka laski kunkin havainnon poikkeaman pankkikohtaisesta keskiarvosta ja keskihajonnasta. Poikkeaviksi määriteltiin ne siirrot, joiden z-arvon itseisarvo ylitti arvon 2. Tämä malli toimi erityisesti vertailupohjana koneoppimismalleille ja tarjosi helposti tulkittavan menetelmän havaita anomalioita. (Alpaydın, 2020, s 41.)

Varsinaisten mallien toteutus tapahtui `scikit-learn`-kirjaston tarjoamien työkalujen avulla. Esimerkiksi poikkeamien tunnistamiseen rakennettiin erillinen Isolation Forest -malli, joka soveltuu erityisesti epänormaalien havaintojen löytämiseen ilman valvottua opetusdataa. Mallin konfiguraatio sisälsi parametrien, kuten n_estimators ja contamination, säätämisen, minkä avulla pyrittiin optimoimaan havaintojen erottelukyky realistisessa liiketoimintaympäristössä. (scikit-learn developers, 2025.) Mallin toimintaperiaate havainnollistuu alla olevassa kuvassa, jossa datapisteet kulkevat useiden iTree-puiden läpi ja poikkeavat havainnot erottuvat lyhyempien polkujen ja korkean anomaly score -arvon perusteella



KUVA 1. Diagrammi Isolation Forest -mallin toiminnasta. Kuvan tuotti OpenAI:n ChatGPT-malli käyttäjän ohjeiden perusteella (13.6.2025). Visualisointi perustuu mallin teoreettiseen kuvaukseen (Explainable Artificial Intelligence for Intelligent Transportation Systems ym., 2023., s. 7).

Kolmantena toteutettiin valvottu luokittelumalli Random Forest Classifier -algoritmilla, joka hyödynsi aiemmin määritettyjä poikkeamamerkintöjä. Mallin tavoitteena oli oppia poikkeamien tunnusmerkit ja luokitella uusia havaintoja todennäköisyysperusteisesti. Esikäsittelyssä kategoriset muuttujat, kuten pankki ja viikonpäivä, muunnettiin OneHotEncoderin avulla numeerisiksi piirteiksi, ja data jaettiin harjoitus- ja testijoukkoihin `train_test_split`-funktion avulla. Mallin toimintaperiaate esitetään alla olevassa kuvassa, jossa data jaetaan bootstrap-näytteisiin, jokainen puu tuottaa oman ennusteensa, ja lopullinen luokka muodostetaan enemmistöäänestyksellä.



KUVA 2. Skeema Random Forest -luokittelu-algoritmin rakenteesta. Kuvan tuotti OpenAI:n CHATGPT-malli käyttäjän ohjeiden perusteella (13.06.2025). Visualisointi perustuu mallin teoreettiseen kuvaukseen (Fu & Qi, 2022; OpenAI, 2025)

Kaikki mallien tulokset kirjoitettiin takaisin BigQuery-tauluihin `load_table_from_dataframe`-toiminnolla, jolloin ne saatiin esitettyä suoraan Grafana-paneeleissa. Tämä mahdollisti poikkeamien, todennäköisyyksien ja mallien tulosten reaaliaikaisen visualisoinnin analytiikka-alustalla.

Python-ympäristöksi valittiin versio 3.10, joka tarjosi vakaat yhteensopivuudet käytettyjen kirjastojen kanssa. Mallien kehitys tehtiin paikallisesti Visual Studio Codessa sekä Google Cloudin hallitussa Vertex AI Workbench -notebook-ympäristössä, jossa oli valmiiksi asennettu tarvittavat koneoppimiskirjastot.

3.1.4 Vertex AI -mallin hyödyntäminen

Vertex AI:n avulla rakennettiin koneoppimiseen perustuva aikasarjaennustemalli, joka ennustaa tulevia rahansiirtosummia pankkien ja yrityksen välillä. Mallin päätavoitteena oli muodostaa perustaso (baseline) tapahtumasummille, johon

verrattiin toteutuneita tapahtumia poikkeamien havaitsemiseksi. Vertex AI mahdollisti skaalautuvan mallinrakennuksen ja automaattisen mallinhallinnan ilman tarvetta ylläpitää omaa infrastruktuuria.

Mallin toteutus aloitettiin määrittämällä ennustemallin syötedata BigQuery-taulusta `rahavalvonta.transactions`. Data aggregoitiin pankki- ja aikatasolla (päivittäin/tunneittain) siten, että jokaiselle pankille muodostui oma aikasarja. Jokaisessa aikasarjassa oli kaksi keskeistä muuttujaa: aikaleima (timestamp) ja siirretty rahamäärä (amount). Tämä data vietiin Vertex AI Forecast -palveluun käyttäen Vertex AI Workbench -ympäristössä kehitettyä Python-koodia.

Mallin koulutuksessa käytettiin `google.cloud.aiplatform`-kirjastoa. Ennustemalli luotiin seuraavin päävaihein:

1. **Datan valmistelu ja lataus**: Datan muotoilussa varmistettiin, että aikasarjojen saraketunniste (time_series_identifier_column) vastasi yksittäistä pankkia, ja ajallinen muuttuja (timestamp_column) noudatti ISO 8601 -muotoa. Rahasummat asetettiin `target_column`-kenttään.

2. **Mallin koulutus Vertex AI Forecast -mallilla**:
   Ennustemalli määriteltiin hyödyntämään automaattisesti parasta algoritmia (AutoML Forecasting). Määritettiin ennustehorisontti (2 tuntia), aikaintervalli (1 tunti) ja valittiin tärkeitä ennustemuuttujia (esim. pankin tyyppi, viikonpäivä).

3. **Mallin käyttöönotto ja tulosten vienti BigQueryyn**: Malli ajettiin testidatalla ja tulokset kirjoitettiin takaisin BigQuery-tauluun `rahavalvonta.predictions`. Näin mallin ennusteet saatiin suoraan käytettäväksi Grafanan visualisointipaneeleissa ilman erillistä välikäsittelyä.

4. **Mallin evaluointi**: Vertex AI tuotti automaattisesti arviointimetriikat (MAE, RMSE), jotka tallennettiin erilliseen `rahavalvonta.model_evaluation`-tauluun. Näitä tuloksia hyödynnettiin Grafanan mallikohtaisessa "mallin laatu" -paneelissa.

Vertex AI Forecastin käyttö mahdollisti tehokkaan ja kustannustehokkaan ennustemallin toteutuksen ilman, että manuaalista koneoppimiskoodia tarvitsi kirjoittaa alusta asti. Mallin konfiguraatiot pysyivät läpinäkyvinä, ja ennustetarkkuus oli useissa testipankeissa riittävällä tasolla, erityisesti arkiaamujen rutiiniryppäissä (*Google*, 2025c).

3.1.5 Datan simulaation suunnittelu

Todellisten pankkisiirtojen tai maksuliikennetapahtumien hyödyntäminen ei ollut mahdollista, sillä data on sekä luottamuksellista että liiketoimintakriittistä. Näin ollen päädyttiin (tuotantoa vastaavan) simuloidun datan tuottamiseen, jonka avulla oli mahdollista toteuttaa tutkimukselliset tavoitteet ilman tietosuojaan tai käyttöoikeuksiin liittyviä rajoitteita.

Simuloidun datan suunnittelussa tavoiteltiin mahdollisimman todenmukaista lopputulosta. Tämä tarkoitti, että aineiston tuli noudattaa rahansiirtojen luontevia ajallisia, määrällisiä ja rakenteellisia piirteitä. Aikasarjamainen rakenne oli erityisen tärkeä, jotta mallit pystyivät oppimaan tapahtumien rytmin ja mahdolliset poikkeamat siihen. Myös pankkikohtaisia eroja mallinnettiin siten, että esimerkiksi OP:lle generoitiin enemmän tapahtumia kuin pienemmille toimijoille kuten OmaSP:lle. Tällä tavalla aineistoon muodostui luonnollisia volyymieroja, jotka paransivat mallien kykyä tunnistaa epänormaalia käyttäytymistä.

Simulaatio rakennettiin Pythonin avulla, käyttäen työkaluina muun muassa `pandas`, `random` ja `faker`-kirjastoja. Tapahtumat luotiin kolmivuotiselle ajanjaksolle, kattaen vuodet 2023–2025. Jokaista päivää kohden tuotettiin vaihteleva määrä maksutapahtumia – arkipäivinä erityisesti klo 8–22 välillä, ja viikonloppuisin korostetusti ilta-aikaan. Tällä mallinnettiin verkkokaupan käyttöhuippuja, jotka painottuvat vapaa-aikaan ja ostosten tekemiseen iltaisin esimerkiksi perjantaisin ja lauantaisin. Aineistoon sisällytettiin lisäksi satunnaisia poikkeamia, kuten äkillisiä kasvupiikkejä tai kohonneita epäonnistumisprosentteja, joiden avulla testattiin mallien herkkyyttä ja kykyä tunnistaa epänormaalia käyttäytymistä.

Lopullinen data tallennettiin kuukausittaisina CSV-tiedostoina, jotka ladattiin Google BigQuery -tietokantaan. Datan latausvaiheessa varmistettiin, että kenttien tyypitys, aikaleimaformaatti ja tietojen eheys säilyivät. Taulu `rahavalvonta.transactions` toimi koko analytiikkaputken perusaineistona, ja sen päälle rakennettiin sekä esikäsittelykerrokset että koneoppimismallit. BigQuery valikoitui työkaluksi erityisesti sen suorituskyvyn, skaalautuvuuden ja suoran yhteensopivuuden Grafanan kanssa.

TAULUKKO 1. Simuloidun datan kenttärakenne

Sarake	**Tyyppi**	**Kuvaus**
transaction_id	Teksti (UUID)	Yksilöllinen tunniste jokaiselle tapahtumalle
timestamp	Aikaleima	Tapahtuman ajankohta (UTC); käytetään aikasarjamallinnukseen
bank	Teksti	Pankin nimi
amount	Desimaali	Rahansiirron summa euroissa
status	Teksti	Tapahtuman tila
type	Teksti	Tapahtuman tyyppi

Simulointi osoittautui paitsi teknisesti onnistuneeksi, myös oppimisen kannalta hyödylliseksi prosessiksi. Sen kautta oli mahdollista hahmottaa, millaiset ilmiöt erottuvat datassa ajallisesti, kuinka paljon variaatiota kuuluu normaaliin toimintaan, ja millaisissa tilanteissa järjestelmä voisi aidosti tarvita valvontaa. Vaikka data ei perustunut oikeaan maksuliikenteeseen, sen rakenne ja dynamiikka loivat pohjan uskottavalle mallinnukselle ja visualisoinnille.

TAULUKKO 2. Keskimääräiset tapahtumat / päivä

Viikonpäivä	**Keskimääräiset tapahtumat/päivä**	**Kommentti**
Maanantai	2200	Korkea aktiivisuus työviikon alussa
Tiistai	2400	Tasainen arkipäivä
Keskiviikko	2300	Keskimääräinen volyymipäivä
Torstai	2500	Usein vilkkain päivä
Perjantai	2700	Vilkasta etenkin iltaisin
Lauantai	1800	Iltaisin piikki verkkokaupassa
Sunnuntai	1600	Myöhäisillan tapahtumia korostettu

TAULUKKO 3. Keskimääräinen osuus tapahtumista

Pankki	**Keskimääräinen osuus tapahtumista (%)**	**Kommentti**
OP	32 %	Suurin yksittäinen toimija
Nordea	26 %	Laaja asiakaskunta
Danske Bank	18 %	Aktiivinen erityisesti arkipäivisin
OmaSP	12 %	Pienempi volyymi, tasainen rytmi
S-Pankki	12 %	Painottuu enemmän viikonloppuihin

3.1.6 Tiedonsiirron rajapinnat

Tietojen siirtäminen järjestelmien välillä oli keskeinen osa koko koneoppimiseen perustuvan valvontaratkaisun toiminnallisuutta. Koska simuloitu data tuotettiin erillisellä Python-skriptillä ja tallennettiin Google BigQuery -tietovarastoon, tuli rajapintojen kautta tapahtuvasta tiedonsiirrosta järjestelmän selkäranka. Rajapinnat mahdollistivat toisaalta datan jatkuvan päivittymisen mallien koulutusta ja ennustamista varten, ja toisaalta tulosten siirtämisen takaisin visualisoitavaan muotoon.

Ratkaisussa hyödynnettiin ensisijaisesti Google Cloudin sisäisiä komponentteja, kuten BigQueryn SQL-rajapintaa, Vertex AI:n mallien hallintaa sekä Python-pohjaisten skriptien integrointia Google Cloud Workbenchiin. Mallien koulutuksessa ja ennustamisessa käytettiin joko suoraa yhteyttä BigQueryyn (`google.cloud.bigquery` -kirjasto) tai Vertex AI Forecast -mallien tapauksessa automatisoitua datan viennin ja tulosten palauttamisen prosessia.

Tiedonsiirto perustui selkeään jaettuun tietomalliin: kaikki tapahtumadata, mallitulokset ja poikkeamahavainnot siirrettiin BigQuery-tauluihin. Jokainen mallityyppi kirjoitti tuloksensa omaan tauluunsa (`anomaly_iforest_results`, `predictions_forecast`, `classification_results`), jolloin mallit eivät sekoittuneet keskenään ja tulosten versiointi oli yksiselitteistä.

Kun tulokset olivat saatavilla BigQueryssä, ne voitiin hakea Grafanaan joko suoraan kyselyiden kautta tai valmiiksi käsiteltyinä näkyminä (views). Tämä mahdollisti myös suodatuksen ja dynaamisen valinnan käyttäjän toimesta, esimerkiksi tietyn pankin, mallityypin tai ajanjakson mukaan. Grafana toimi tässä kontekstissa puhtaasti lukijana, eikä siihen tarvinnut rakentaa erillistä backend-logiikkaa. Kaikki logiikka ja data pysyivät Google Cloud -ekosysteemissä, mikä yksinkertaisti kokonaisarkkitehtuuria ja paransi tietoturvaa.

3.1.7 Visualisointikerroksen suunnittelu (Grafana Cloud)

Visualisointikerros suunniteltiin toteutettavaksi Grafana Cloud -ympäristössä, joka tarjosi valmiin, skaalautuvan ja selainkäyttöisen alustan koneoppimismallien tulosten ja rahansiirtodatan reaaliaikaiseen esittämiseen. Valintaan vaikutti keskeisesti se, että Grafana oli jo entuudestaan käytössä organisaation muussa järjestelmävalvonnassa, minkä vuoksi sen hyödyntäminen oli luontevaa ja mahdollisti nopean käyttöönoton ilman tarvetta kartoittaa vaihtoehtoisia ratkaisuja. Lisäksi organisaatiossa oli ennestään osaamista Grafanan konfiguroinnista, liitännöistä ja visualisointien rakentamisesta, mikä alensi oppimiskynnystä ja nopeutti kehitystä.

Grafanan käyttö mahdollisti tietolähteiden yhdistämisen suoraan BigQueryyn, jolloin mallien tulokset, transaktiodata ja mahdolliset poikkeamat saatiin esitettyä ilman välikerroksia tai ylimääräisiä sovelluksia (*Grafana Labs, 2025a.*) Visualisointien perustaksi valittiin SQL-pohjaiset kyselyt, jotka suoritettiin BigQueryssa. Tämä mahdollisti sen, että Grafana toimi puhtaasti visualisointikerroksena, eikä siirtänyt laskentakuormaa palvelimelle tai selaimelle.

Kukin koneoppimismalli sai oman rivinsä dashboardilla, ja sen sisällä oli paneelit mallin versionumerolle, arviointimittareille (esim. MAE, RMSE, Precision, Recall) sekä mallikohtaisille havainnoille tai pankkikohtaisille poikkeamatiedoille. Paneelit suunniteltiin tukemaan suodatusta käyttäjän valitsemien mallien, pankkien tai aikajaksojen mukaan. Visuaalinen suunnittelu korosti selkeyttä ja saavutettavuutta, ja kaikki tulokset pyrittiin esittämään siten, että myös ei-tekninen käyttäjä pystyy ymmärtämään mallien havainnot ja seurattavat indikaattorit.

Graafisen käyttöliittymän rakenteessa huomioitiin käyttäjän lukusuunta ja looginen eteneminen vasemmalta oikealle sekä ylhäältä alas. Visualisointeihin lisättiin värikoodatut statusboksit, viivagraafit ja sarakevertailut, joilla eri mallien ja pankkien suorituskyky saatiin helposti vertailtavaan muotoon. Suunnittelu pohjautui myös siihen, että dashboardia voidaan myöhemmin laajentaa

esimerkiksi lisäämällä uusia malleja tai käyttämällä toisenlaisia visualisointikomponentteja.

Koska kyseessä oli prototyyppivaihe, dashboardiin ei rakennettu loppukäyttäjien kirjautumista, roolipohjaisia näkymiä tai interaktiivisia hälytysjärjestelmiä. Hälytykset esitettiin visuaalisina merkkeinä, kuten väreinä tai poikkeamakuvakkeina, eikä niitä kytketty sähköposti- tai webhook-ilmoituksiin (*Grafana Labs*, 2025b.). Graafisen käyttöliittymän päätavoitteena oli tarjota järjestelmälle selkeä ja luotettava tulkintapinta ilman teknistä monimutkaisuutta tai raskasta hallintamallia. Näin rakennettu visualisointikerros tukee järjestelmän käytettävyyttä ja mahdollistaa skaalautuvuuden sekä tulevat kehitysvaiheet.

3.2 Arkkitehtuurisuunnitelma

Valvontaratkaisun arkkitehtuuri suunniteltiin tukemaan koneoppimismallien käytännön soveltamista pankkisiirtojen poikkeamien havaitsemisessa. Rakenteen suunnittelussa pyrittiin siihen, että järjestelmä olisi selkeästi jäsennelty, skaalautuva ja helposti laajennettavissa myös tilanteisiin, joissa lähdedata on tuotantokäyttöön tuotua reaaliaikaista aineistoa. Koska työ toteutettiin simuloidulla datalla, arkkitehtuuri rakennettiin modulaariseksi, jolloin yksittäiset osat, kuten datasyötteet, mallit tai visualisointikerros voitaisiin myöhemmin korvata tai laajentaa ilman, että koko kokonaisuus vaatisi uudelleenrakennusta.

Arkkitehtuurin suunnittelussa painotettiin sitä, että ratkaisu perustuu avoimiin ja helposti hallittaviin työkaluihin. Tämän vuoksi valitut keskeiset komponentit olivat Python-pohjaiset mallinnusskriptit, Google BigQuery tietovarastona ja Grafana visualisointialustana. Näiden varaan rakennettiin valtaosa toteutuksesta. Ainoastaan yksi neljästä mallista, ennustamiseen perustuva Forecast-malli, toteutettiin Vertex AI:n AutoML Forecasting -ominaisuutta hyödyntäen, koska se tarjosi tehokkaan ja helposti käyttöönotettavan tavan toteuttaa monimutkainen aikasarjaennustemalli ilman erillistä ohjelmointia. Kaikki muut mallit – Z-score, Isolation Forest ja luokittelumalli – rakennettiin ja ajettiin Pythonilla, mikä

mahdollisti täyden kontrollin logiikasta, versiopäivityksistä ja mallien säätämisestä.

Keskitetty tietovarasto rakennettiin BigQueryn varaan, mutta se ei toiminut pelkästään datan säilytyspaikkana. Suunnittelussa huomioitiin, että BigQuery mahdollistaa samassa ympäristössä sekä data-analyysin että suorat mallien syötteet ja tulosten hakemisen visualisointiin. Näin vältettiin ylimääräiset siirtovaiheet ja mahdolliset kopioinnista johtuvat tietojen ristiriidat. Arkkitehtuurisesti tämä tarkoitti sitä, että kaikki data- ja mallitaulut rakennettiin yhteiseen skeemaan, jonka sisällä oli johdonmukainen nimeäminen, aikaleimapohjainen järjestäminen ja selkeä versiohallintarakenne.

Pythonilla toteutetut mallit suoritettiin Google Cloudin Workbench-ympäristössä, jossa käytettiin valmiita kirjastoja, kuten `pandas`, `sklearn` ja `google.cloud.bigquery`. Tämä mahdollisti suoran integraation tietovaraston kanssa. Ennustemallissa käytetty Vertex AI Forecast integroitiin natiivisti BigQueryyn, jolloin mallin koulutus ja ajaminen voitiin automatisoida Vertexin käyttöliittymän ja ajastettujen työnkulkujen kautta ilman erillistä koodia. Yhdistämällä Python-malleja silloin kun tarvitaan mukautettavuutta ja Vertex AI -alustaa silloin kun ennustemalli on monimutkainen mutta vakioitavissa, saavutettiin teknisesti tehokas ja hallittu kokonaisuus.

Arkkitehtuurin suunnittelussa huomioitiin myös mallien tulosten esittäminen ei-teknisille käyttäjille. Tämän vuoksi Grafana visualisointialustana irrotettiin täysin mallien logiikasta, ja se toimi ainoastaan lukijana BigQuerystä. Visualisointiin ei rakennettu omaa backend-palvelua, vaan kaikki logiikka oli toteutettu datan tasolla: mallien tulokset oli tallennettu valmiiksi käyttökelpoisessa muodossa, jolloin Grafanan paneelit pystyivät lukemaan ne suoraan SQL-kyselyillä ilman lisämuotoilua tai käsittelyä.

Kokonaisuutena arkkitehtuuri rakentui teknisesti hallittavaksi kokonaisuudeksi, jossa painopiste oli Pythonilla toteutettavissa, avoimissa ja läpinäkyvissä ratkaisuissa. Vertex AI hyödynnettiin vain silloin, kun sen tarjoamat kyvykkyydet toivat lisäarvoa suhteessa Pythonin joustavuuteen. Tämä lähestymistapa tuki hyvin tutkimusluontoista kehitystyötä ja loi pohjan skaalautuvalle

tuotantoratkaisulle, jossa eri komponentit voivat kehittyä omaan tahtiinsa ilman riippuvuuksia toisiinsa.

3.3 Kustannusten hallinta suunnittelussa

Yritysten tuotantoympäristöissä järjestelmän suorituskyky, luotettavuus ja laajennettavuus menevät usein kustannusten edelle, etenkin kun kyse on liiketoimintakriittisestä valvontaratkaisusta. Tällöin investoinnit skaalautuviin palveluihin, automaatioon ja tietoturvaan voidaan perustella suoraan liiketoimintahyödyillä. Opinnäytetyössä tilanne oli kuitenkin toinen: tavoitteena oli toteuttaa uskottava ja teknisesti toimiva järjestelmä siten, että käytetyt palvelut ja teknologiat pysyivät kohtuullisissa kustannusraameissa opiskelijan käytettävissä olevien resurssien puitteissa.

Tämän vuoksi järjestelmäsuunnittelussa tehtiin harkittuja kompromisseja, joiden avulla kaikki keskeiset toiminnallisuudet voitiin toteuttaa ilman, että järjestelmän laatu tai uskottavuus kärsivät. Esimerkiksi Google Cloud Platformin valinta perustui sen tarjoamiin valmiisiin koneoppimispalveluihin, skaalautuvaan tietovarastointiin ja ilmaiseen aloitus kokeiluun, joka mahdollisti mallien koulutuksen ja tietojen analysoinnin ilman välittömiä lisäkuluja. Silti suunnittelussa vältettiin jatkuvaa tausta-ajoa, automaattisia uudelleenkoulutuksia tai suuria kyselymääriä BigQueryssa, jotta kustannuksia voitiin hallita ennakoitavasti.

Grafana Cloudin osalta valittiin maksullinen mutta kohtuuhintainen Pro-versio, koska se sisälsi BigQuery-integraation, pidemmän datan säilytysajan ja valmiit visualisointitoiminnot ilman tarvetta pystyttää omaa palvelininfrastruktuuria. Käyttöönotto ja hallinta pysyivät kevyinä, mutta samalla dashboardien laatua ei tarvinnut heikentää. Toisaalta päätettiin jättää pois roolipohjaiset käyttäjähallinnat, loppukäyttäjien kirjautumistoiminnot ja automatisoidut hälytykset sähköpostiin tai ulkoisiin järjestelmiin – nämä olisivat lisänneet projektin kompleksisuutta ja mahdollisesti vaatineet laajempia lisenssejä tai kolmannen osapuolen palveluita.

Python-ympäristö toteutettiin avoimen lähdekoodin periaattein: venv-pohjainen virtuaaliympäristö ja requirements.txt-tiedosto mahdollistivat hallitun ja siirrettävän kehitysalustan ilman kustannuksia. Kehitysalustavaihtoehdoista suosittiin Vertex AI Workbenchiä, kun se oli saatavilla osana GCP:n ekosysteemiä, mutta vaihtoehtona huomioitiin myös paikallinen Notebook-ympäristö.

Vaikka kaikki toiminnot saatiin toteutettua käytettävissä olevilla työkaluilla, suunnittelussa painotettiin erityisesti sitä, että ratkaisut olisivat laajennettavissa tuotantotason käyttöön ilman, että koko arkkitehtuuria täytyisi muuttaa. Toisin sanoen kustannustehokkuus ei tarkoittanut laadusta tinkimistä, vaan valintoja, joissa säilyi tekninen uskottavuus ja todelliseen käyttöön soveltuva rakenne. Tämä lähestymistapa kuvastaa hyvin sitä, kuinka myös rajatuin resurssein voidaan suunnitella järjestelmä, joka on aidosti käyttökelpoinen ja helposti jalkautettavissa laajempiin ympäristöihin.

4 TOTEUTUKSEN TEKNINEN LÄPIVIENTI

4.1 Kehitysympäristö ja GCP-infrastuktuurin käyttöönotto

Projektin kehitystyö käynnistettiin luomalla uusi Google Cloud Platform -projekti, jonka sisälle aktivoitiin vain ne palvelut, joita tarvittiin varsinaiseen toteutukseen. Ensisijainen tavoite oli rakentaa mahdollisimman kevyt mutta toimiva infrastruktuuri, joka mahdollisti koneoppimismallien kehityksen, datan käsittelyn ja tulosten visualisoinnin keskitetysti samassa ympäristössä.

GCP:n selainkäyttöliittymän kautta otettiin käyttöön erityisesti Cloud Storage tiedostojen säilytystä varten sekä Vertex AI Workbench, jota käytettiin Python-pohjaisena kehitysalustana. Vertex AI Workbench tarjosi Jupyter-notebook-tyyppisen käyttöliittymän, jossa kehitys, analysointi ja mallinnus voitiin suorittaa suoraviivaisesti pilvessä. Työssä käytetty instanssityyppi oli n1-standard-4, joka tarjosi 4 virtuaaliprosessoria ja 15 gigatavua keskusmuistia, joka oli riittävä kokoonpano suunnitelluille tehtäville.

Kehitysympäristöön asennettiin tarvittavat Python-kirjastot (kuten `pandas`, `numpy`, `matplotlib`, `scikit-learn`, `google-cloud-storage`), ja yhteys Cloud Storageen sekä muihin GCP-palveluihin varmistettiin testiskripteillä. Projektin rajaukseen kuului, ettei kehitystyössä hyödynnetty erillistä CI/CD-järjestelmää, Dockeria tai orkestrointityökaluja – ratkaisu pidettiin tarkoituksella yksinkertaisena ja helposti hallittavana.

Simuloitu aineisto tuotettiin paikallisesti kuukausittaisiksi CSV-tiedostoiksi, jotka siirrettiin Cloud Storage -palveluun. Tiedostot jäsenneltiin kansiorakenteeseen rahavalvonta/csv/, ja tiedostonimien muoto noudatti rakennetta 2023_01.csv, 2023_02.csv jne. Yhteensä palveluun ladattiin 36 CSV-tiedostoa, kattaen kuukaudet tammikuusta 2023 joulukuuhun 2025. Tämä kansiorakenne muodosti projektin datapohjan, jota käsiteltiin Pythonilla suoraan pilvessä ilman tarvetta siirrellä tiedostoja paikallisesti.

Notebook-ympäristö integroitiin suoraan Cloud Storageen niin, että tiedostoja voitiin lukea ja käsitellä Pythonin `gcsfs`- ja `google.cloud.storage`-kirjastojen avulla. Tämä ratkaisu teki kehitystyöstä tehokasta, koska eri työvaiheita voitiin toteuttaa samassa ympäristössä ilman siirtymiä muiden kehitysalustojen välillä. Myös tulosten esikatselu, mallien testiajo ja tiedostojen tarkastus tapahtuivat kaikki yhdessä käyttöliittymässä.

Kaiken kaikkiaan kehitysympäristö ja infrastruktuuri pysyivät koko toteutusvaiheen ajan kevyenä, hallittavana ja vakaana. Yhtenäisen projektin ja vähäisen palvelukokonaisuuden ansiosta ympäristön ylläpito ja mahdollisten virheiden paikantaminen oli suoraviivaista. Tämä muodosti hyvän perustan mallien toteutukselle, jotka käsitellään tarkemmin seuraavissa kappaleissa.

4.2 Datan lataus BigQueryyn

Cloud Storageen siirretty simuloitu CSV-aineisto ladattiin Google BigQuery -tietovarastoon, joka toimi mallien harjoitus- ja tulostaulujen pääasiallisena säilytyspaikkana. Latausprosessi suunniteltiin yksinkertaiseksi ja toistettavaksi, jotta aineistoa voitiin päivittää tarpeen mukaan ilman ylimääräistä käsityötä.

Simuloitu aineisto tuotettiin Pythonilla Visual Studio Code -kehitysympäristössä, jossa skripti muodosti kuukausittaiset CSV-tiedostot oikeassa muodossa ja tallensi ne paikalliseen kansiorakenteeseen. Valmiit CSV-tiedostot siirrettiin tämän jälkeen Google Cloud Storageen, josta ne olivat valmiina ladattaviksi BigQueryyn.

Varsinainen lataus BigQueryyn toteutettiin pääosin BigQueryn selainkäyttöliittymän ja Python-skriptien yhdistelmällä. Ensimmäisessä vaiheessa luotiin pysyvä taulu `rahavalvonta.transactions`, jonka skeema määriteltiin manuaalisesti vastaamaan CSV-tiedostojen sarakkeita. Skeemassa huomioitiin erityisesti timestamp-sarakkeen aikavyöhyke (Europe/Helsinki) sekä amount-kentän tarkkuus (FLOAT64). Jokainen rivi sisälsi tietoja yksittäisestä tapahtumasta, kuten siirtopäivämäärän, summan, pankin nimen ja maksualustan.

CSV-tiedostot ladattiin BigQueryyn suoraan Cloud Storagesta käyttäen joko bq-komentorivityökalua tai Pythonin `google-cloud-bigquery`-kirjastoa. Kehitetty Python-skripti mahdollisti koko 36 tiedoston eräajon yhdellä ajokerralla, mikä nopeutti alkuperäisen aineiston siirtoa tuotantoympäristöön. Jokainen tiedosto ladattiin append-tilassa samaan `transactions`-tauluun, jolloin datan jakaminen kuukausiksi ei vaikuttanut analysoitavuuteen.

Datan latausprosessin aikana suoritettiin kevyitä tarkistuksia, kuten rivimäärän vertaaminen odotettuun ja puuttuvien kenttien havaitseminen. Aikamuotojen yhteensopivuus varmistettiin erityisesti timestamp-kentässä, jotta mallien käyttämät suodattimet toimivat oikein myös aikaväleittäin.

Yksi tärkeimmistä päätöksistä oli se, että alkuperäinen CSV-aineisto säilytettiin Cloud Storagessa, vaikka kaikki analyysi tapahtui BigQueryssä. Tämä mahdollisti latausten uusimisen ilman datan rekonstruointia ja loi perustan versionhallittavalle, dokumentoidulle tietoputkelle.

4.3 Mallikohtainen implementaatio

Mallien toteutus keskittyi ennen kaikkea eriytettyyn ja käytännönläheiseen kehitykseen, jossa kunkin mallin logiikka rakennettiin itsenäisesti testattavaksi ja ajettavaksi kokonaisuudeksi. Tekemisen painopiste oli siinä, että jokainen malli voitiin toteuttaa ilman ulkoisia riippuvuuksia – myös niiden kokeilu, virheenkäsittely ja versiointi pysyivät yksinkertaisina.

Toteutusprosessissa ei hyödynnetty valmista mallirunkoa tai automatisoitua putkea, vaan työ eteni suoraviivaisesti: datan haun jälkeen malli rakennettiin suoraan analyysitarpeeseen sopivaksi. Tämä antoi vapautta valita lähestymistapa mallikohtaisesti ja mahdollisti nopean etenemisen eri mallityyppien välillä.

Käytännössä tämä tarkoitti myös sitä, että mallien toteutus ei ollut teknisesti raskasrakenteinen vaan ratkaisut kehitettiin suoraan käytettävyyttä ja läpinäkyvyyttä painottaen. Tiedot tallennettiin mallikohtaisiin tauluihin ilman

integraatiopainetta, ja visualisointi oli helppo toteuttaa, kun datan muoto pysyi hallitusti vakiona.

Toteutusvaiheessa keskeistä oli, että mallin sai helposti käyntiin, tarkistettua ja tarvittaessa ajettua uudelleen muuttamalla vain parametreja – ilman tarvetta muokata koko mallilogikkaa. Tämä lähestymistapa osoittautui toimivaksi erityisesti usean mallin rinnakkaisessa kehityksessä.

4.3.1 Z-Score: perusanalyysi ja hälytystaso

Z-score-mallin toteutus toimi projektin ensimmäisenä teknisenä kokonaisuutena, jonka avulla testattiin koko analytiikkaputken toimivuutta alusta loppuun. Malli perustui yksinkertaiseen tilastolliseen lähestymistapaan, jossa pankkikohtaiset rahansiirrot standardoitiin suhteessa niiden keskiarvoon ja keskihajontaan. Mallin tavoitteena oli tunnistaa tilanteet, joissa tietyn aikajakson siirtomäärä poikkeaa merkittävästi historiallisesta vaihtelusta.

Datan haku suoritettiin BigQuerysta käyttäen SQL-kyselyä, joka aggregoi tapahtumat pankin ja päivän mukaan. Tiedot tuotiin Pandas DataFrameen, missä laskettiin z-score-arvot kaavalla:

$$Z = \frac{x - \mu}{\sigma}$$

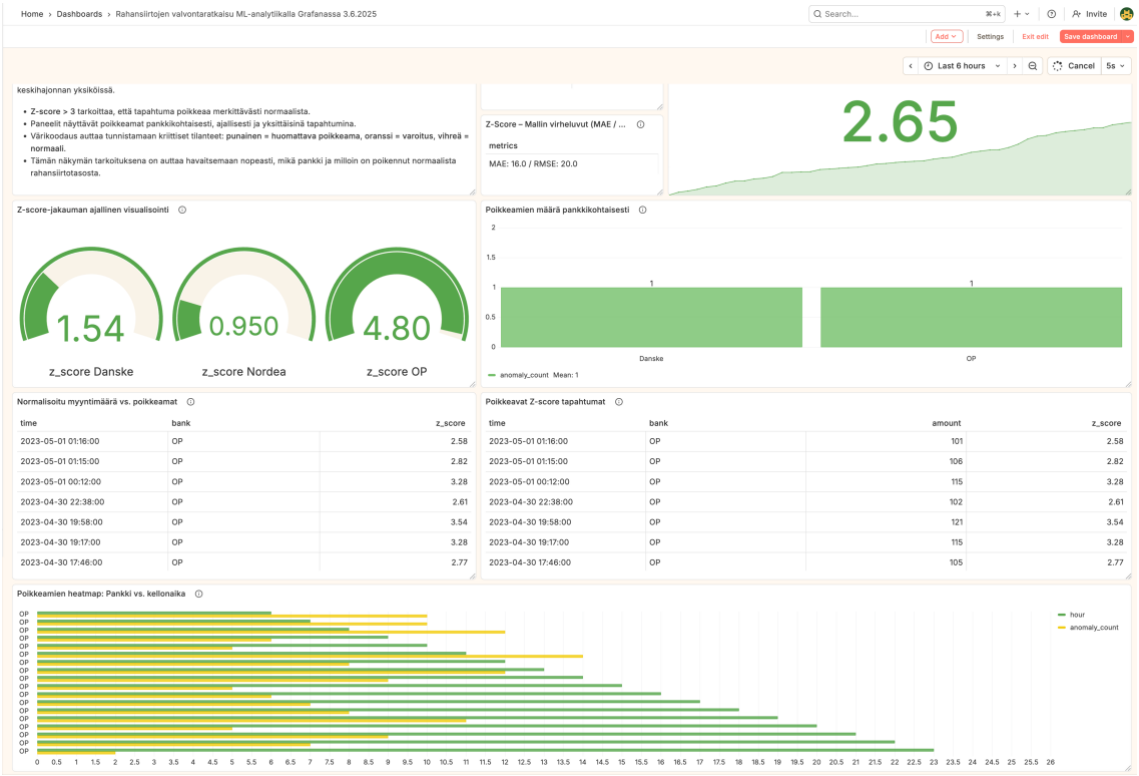
missä x on päivän rahasumma, μ historiallinen keskiarvo ja σ keskihajonta (James ym., 2013, luku 2.3 Feature Scalling).

Mallissa käytettiin liikkuvaa 7 päivän ikkunaa keskiarvon ja keskihajonnan laskemiseen, jolloin analyysi reagoi ajan myötä muuttuviin tasoihin mutta ei ylireagoinut yksittäisiin poikkeamiin. Hälytysrajat asetettiin ehdollisesti: mikäli päivän Z-arvo oli yli +2 tai alle -2, rivi merkittiin poikkeamaksi.

Mallin tulokset sisälsivät sekä alkuperäisen summan että laskettuja tunnuslukuja: Z-arvon, rajan ylityksen ja poikkeamalipun (anomaly_flag). Tiedot kirjoitettiin

mallikohtaiseen BigQuery-tauluun `rahavalvonta.zscore_results`, jonka rakenne mahdollisti helpon jatkokäytön Grafanassa ja mallien vertailussa.

Z-score-malli oli nopea toteuttaa, skaalautuva eri pankkeihin ja tarjosi läpinäkyvän analyysimenetelmän, jonka tulokset olivat helposti tulkittavissa myös ei-teknisille sidosryhmille. Vaikka menetelmä ei huomioi rakenteellisempia muutoksia (esim. viikonloppuvaikutuksia tai kausivaihtelua), se osoittautui käyttökelpoiseksi lähtötason hälytysjärjestelmäksi, johon voitiin verrata muiden mallien antamia poikkeamatuloksia.



KUVA 3. Grafanan Z-score-näkymä pankkivalvontaan liittyvistä poikkeamista viimeisen 30 päivän ajalta.

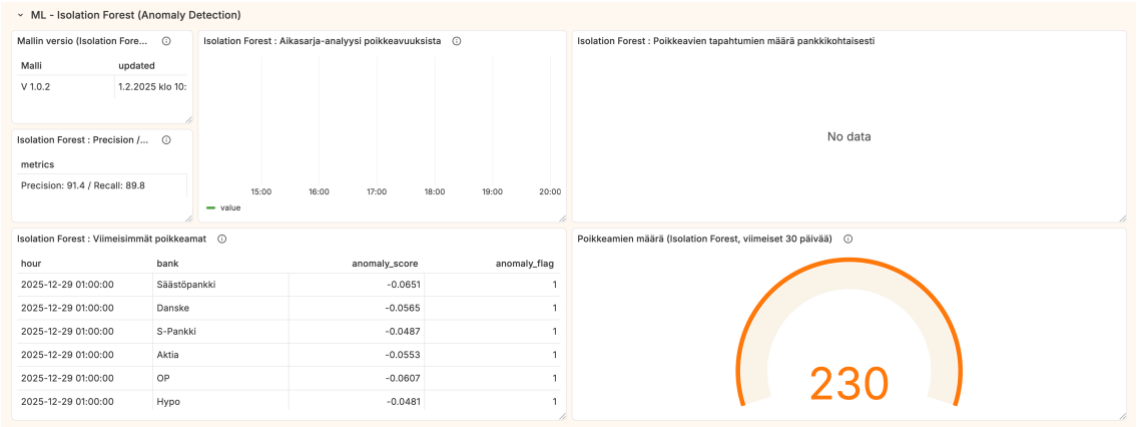
4.3.2 Isolation Forest: piirreanalyysi ja käyttö

Isolation Forest -malli toteutettiin Pythonilla käyttäen Visual Studio Code -kehitysympäristöä. Mallin tarkoituksena oli havaita epänormaaleja rahansiirtokäyttäytymisen muotoja ilman opetusdataa – eli valvomattomalla (unsupervised) oppimisella. Algoritmin lähtökohtana oli oletus, että poikkeavat havainnot on helpompi eristää muista lyhyemmillä jakopoluilla päätöspuissa.

Mallin lähtöaineistona käytettiin Google Cloud Storageen tallennettua simuloitua CSV-dataa, joka luettiin Pandasin DataFrameksi analyysia varten. Datan esikäsittelyssä muodostettiin useita piirteitä, kuten viikonpäivä, kellonaika, pankin aiempi keskiarvo sekä suhteellinen siirtosumma. Näiden pohjalta rakennettiin mallin tarvitsemat piirrejoukot.

Varsinainen IsolationForest-malli rakennettiin `sklearn.ensemble`-kirjastolla. Mallin oppimisprosessissa käytettiin oletuksena 100 estimointipuuta sekä kontaminaatioarvoa, jonka perusteella arvioitiin poikkeamien osuus aineistossa. Malli sovellettiin koko simuloituun aineistoon ilman valvottua opetusvaihetta (scikit-learn developers, 2025)

Tuloksena saatiin jokaiselle riville mallin laskema `anomaly_score` sekä binäärinen `anomaly_flag`, joka ilmaisi poikkeamien todennäköisyyden. Tulokset kirjoitettiin omaan BigQuery-tauluun `rahavalvonta.anomaly_iforest_results`, josta ne olivat myöhemmin hyödynnettävissä visualisointia ja mallien rinnakkaista käyttöä varten.



KUVA 4. Grafanan näkymä Isolation Forest -mallin tunnistamista poikkeamista pankkien ja yrityksen välisissä rahansiirroissa. Näkymä perustuu viimeisen 30 päivän dataan ja esittää mallin havaitsemat poikkeamat, aikasarjatrendin sekä pankkikohtaisen jakauman.

4.3.3 Vertex AI Forecasting: aikasarjaennuste ja hyödyntäminen

Aikasarjaennustemalli toteutettiin Google Cloudin Vertex AI Forecasting -palvelun avulla hyödyntäen Custom Training -lähestymistapaa. Tavoitteena oli tuottaa pankkikohtaisia lyhyen aikavälin ennusteita rahansiirroista, joita voitaisiin hyödyntää jatkossa esimerkiksi poikkeamien arvioinnissa. Malli valittiin projektin osaksi sen skaalautuvuuden ja integraatiokyvyn vuoksi, mutta toisin kuin täysin automatisoidussa AutoML-mallinnuksessa, tässä haluttiin itse vaikuttaa mallin rakenteeseen ja laskentaympäristöön.

Data ennustemallin pohjaksi haettiin BigQuerystä ja muotoiltiin Vertex AI Forecastingin vaatimusten mukaisesti. Aineisto sisälsi sarakkeet timestamp (UTC-aika), bank (pankkitunniste) ja amount (siirtosumma). Jokainen pankki muodosti oman aikajanan (ts. time series), ja mallille määritettiin seuraavat keskeiset asetukset:

- Time column: timestamp
- Target column: amount
- Time series identifier: bank
- Data granularity: daily
- Forecast horizon: 7 päivää
- Context window: 30 päivää

Custom Training -malli koulutettiin Vertex AI:n hallinnoimassa infrastruktuurissa käyttäen standard-sarjan laskentakoneita. Näin voitiin ohjata resurssien käyttöä ja koulutuksen rakennetta manuaalisemmin kuin AutoML-lähestymistavassa. Koulutusvaiheessa hyödynnettiin yhteensä noin 10 node-tuntia laskenta-aikaa, mikä osoittautui riittäväksi projektin tarpeisiin. Tämä ratkaisu mahdollisti tehokkaan mallin koulutuksen sekä tarkemman hallinnan mallin toimintaan mahdollisissa jatkokehitysvaiheissa.

Mallin valmistuttua se julkaistiin Vertex AI -ympäristössä, ja sen ennustetulokset siirrettiin automaattisesti BigQuery-tietokantaan erilliseen `predictions`-tauluun. Tuloksiin sisältyivät mallin ennustamat euromääräiset summat jokaiselle pankille 7 päivän ajanjaksolle sekä mallin laskemat luottamusvälit. Jokaiselle ennusteelle

tallennettiin myös tekninen tunniste (prediction_id) myöhempää yksilöintiä ja käyttöä varten.

Mallin tuottamien ennusteiden hyödyntäminen Grafana-valvonnassa toteutettiin uudella lähestymistavalla: ennustetulokset luetaan Grafanaan reaaliaikaisesti Prometheus-pohjaisena mittaridatana ilman suoria SQL-kyselyitä. Käytännössä BigQueryyn tallennetut ennusteet muunnetaan automaattisesti monitorointimittareiksi, joita Grafana seuraa jatkuvasti. Tämän ansiosta valvojat näkevät aina ajantasaiset ennusteet rinnakkain toteutuneiden aikasarjojen kanssa viiveettä, eikä Grafana-paneelien tarvitse suorittaa erillisiä kyselyjä BigQueryyn. Ratkaisu poikkeaa aiemmista malleista (Z-score, Isolation Forest, luokittelumalli), joissa Grafana haki tulokset suoraan BigQuery-tauluista SQL-kyselyillä jokaisen päivityksen yhteydessä. Vertex AI Forecasting -mallin kohdalla ennustetieto integroituu suoraan mittarivirtaan, mikä yksinkertaistaa arkkitehtuuria ja vähentää riippuvuutta raskaista tietokantakyselyistä. Samalla mallin koulutus- ja käyttöprosessi on pitkälti automatisoitu: historiallinen data haetaan ja valmistellaan BigQueryssä, malli opetetaan Vertex AI:n ympäristössä, ja ajastettu prosessi generoi säännöllisesti uudet ennusteet BigQueryyn sekä Grafanan luettavaksi. Tämä kokonaisuus varmistaa, että ennustemalli tuottaa jatkuvasti ajantasaista tietoa valvonnan tueksi minimaalisella manuaalisella ylläpidolla.

KUVA 5. Vertex AI Forecasting -mallin ennusteen visualisointi Grafana-näkymässä. Paneeli näyttää pankkikohtaiset rahansiirtojen toteutuneet arvot (historia) sekä mallin ennusteen seuraavalle 7 päivän jaksolle.

4.3.4 Luokittelumalli: koulutus ja käyttöönotto

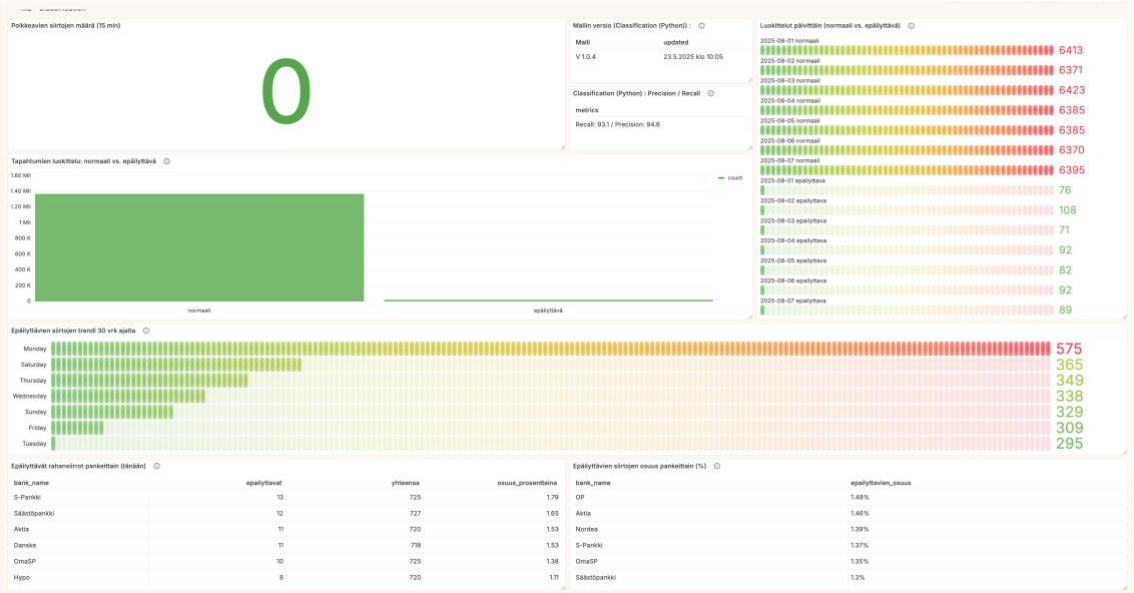
Luokittelumalli rakennettiin Python-ohjelmointikielellä käyttäen Visual Studio Code -kehitysympäristöä. Mallin tarkoituksena oli tunnistaa poikkeavia rahansiirtotapahtumia pankkien ja yrityksen välillä. Toteutus pohjautui valvottuun oppimiseen (supervised learning), jossa käytettiin simuloitua dataa, johon oli lisätty binäärinen risk_flag-sarake osoittamaan poikkeavat tapahtumat.

Mallin kehitystyön yhteydessä dataan johdettiin useita selittäviä muuttujia, kuten viikonpäivä, kellonaika, pankin aiempien tapahtumien keskiarvot ja hajonnat sekä aggregoituja tunnuslukuja eri aikajaksoilta. Datan esikäsittely suoritettiin `Pandas`-kirjastolla ja se jaettiin koulutus-, validointi- ja testijoukkoihin Scikit-learnin train_test_split-toiminnolla.

Luokittelussa käytettiin Gradient Boosting -algoritmia (GradientBoostingClassifier), joka soveltui hyvin epätasapainoiseen aineistoon ja mahdollisti monipuolisen virityksen ilman Vertex AI:n automatisoitua ympäristöä. Mallin koulutus tapahtui Google Cloud -ympäristössä osana Python-skriptiä, joka suoritettiin suunnitellussa työnkulussa muiden mallien rinnalla.

Koulutuksen jälkeen mallia käytettiin arvioimaan uusia tapahtumarivejä, ja jokaiselle riville laskettiin mallin antama todennäköisyysluokka sekä binäärinen ennuste. Tulokset tallennettiin mallikohtaiseen BigQuery-tauluun `rahavalvonta.classification_results`, jolloin ne olivat heti hyödynnettävissä analyysissa ja visualisoinnissa. Tämä mahdollisti yhtenäisen käsittelyn muiden mallien kanssa sekä tulosten versionhallinnan suoraan tietovarastossa.

Mallin kehitys oli ketterä ja käsin ohjattu, mikä antoi paremman kontrollin algoritmivalintoihin ja mallin käyttäytymiseen eri muuttujien osalta. Toteutus soveltui hyvin kokeelliseen ympäristöön ja tarjosi selkeän polun jatkokehitykseen ja mahdolliseen automatisointiin tulevaisuudessa.



KUVA 6. Luokittelumallin tuloksiin perustuvat Grafana-paneelit, jotka visualisoivat normaaleja ja epäilyttäviä rahansiirtoja eri näkökulmista (päivittäin, pankeittain ja viikonpäivittäin).

4.4 Mallien ajastus ja integraatiot

Opinnäytetyössä toteutettujen koneoppimismallien tehokas hyödyntäminen edellytti myös niiden ajastamista, tulosten hallintaa ja järjestelmällistä integrointia muihin osiin kokonaisarkkitehtuuria. Tavoitteena oli rakentaa malli- ja datavirtoja hyödyntävä prosessi, joka toimii ajastetusti ja pystyy päivittämään valvontanäkymän ilman manuaalista väliintuloa. Tämä toteutettiin yhdistämällä Google Cloudin työkalut – erityisesti Vertex AI, Cloud Scheduler, Cloud Functions ja BigQuery – saumattomaksi tietoputkeksi.

Ensimmäisessä vaiheessa määritettiin mallikohtaiset ajoskriptit. Pythonilla toteutetut mallit, kuten Z-Score, Isolation Forest ja luokittelumalli, ajettiin kehitysvaiheessa manuaalisesti Vertex AI Workbenchissä. Näitä malleja ei ajastettu jatkuvaksi, vaan ne suoritettiin tarpeen mukaan, esimerkiksi uuden datan simuloinnin jälkeen. Sen sijaan Vertex AI Forecast -malli integroitiin ajastettuun tuotantoputkeen, jossa sen ennusteita päivitettiin säännöllisesti.

Ajastuksissa hyödynnettiin Cloud Scheduler -palvelua, jonka avulla voitiin määrittää tarkat ajankohdat mallien tai analyysiskriptien käynnistämiseksi.

Näiden ajastettujen tapahtumien yhteydessä aktivoitiin Cloud Functions -toiminto, joka toimi ohjauskerroksena: se käynnisti esimerkiksi BigQuery-kyselyt, mallien uudelleenkoulutuksen tai tietojen kopioinnin ennustetauluihin.

BigQuery toimi integraatiopisteenä kaikille mallien tuloksille. Mallien tuottamat poikkeamaliput ja ennusteet kirjoitettiin tiettyihin tauluihin (`anomaly_zscore`, `anomaly_iforest_results`, `predictions`, `classification_results`), jotka puolestaan toimivat Grafanan näkymien taustana. Koska Grafana hyödynsi suoria kyselyitä BigQuerystä, ei ollut tarvetta välitallennuksille tai tietojen synkronointiin – ajantasainen data näkyi heti visualisoinneissa, kun mallit olivat valmistuneet.

Integraation onnistuminen edellytti myös taulujen rakenteiden yhdenmukaisuutta ja tulosten versiointia. Mallien arviointitiedot tallennettiin `model_evaluation`-tauluun, josta voitiin tarkistaa eri mallien päivitysajankohdat ja suorituskykymittarit (precision, recall, MAE, RMSE jne.). Tällä tavoin saatiin yhtenäinen näkyvyys mallien tilaan ja toimintakuntoon sekä mahdollisuus vertailla niiden toimivuutta valvonnan kannalta.

Koko ajastus- ja integraatioratkaisu rakennettiin vaiheittain, aluksi manuaalisen testauksen kautta ja myöhemmin automatisoiduksi prosessiksi. Tuloksena syntyi teknisesti kevyt, mutta toimiva kokonaisuus, joka osoitti, että koneoppimismallit voidaan yhdistää osaksi valvontaa ilman raskaiden taustajärjestelmien rakentamista – erityisesti silloin, kun käytettävissä on modulaarinen pilviympäristö kuten Google Cloud Platform.

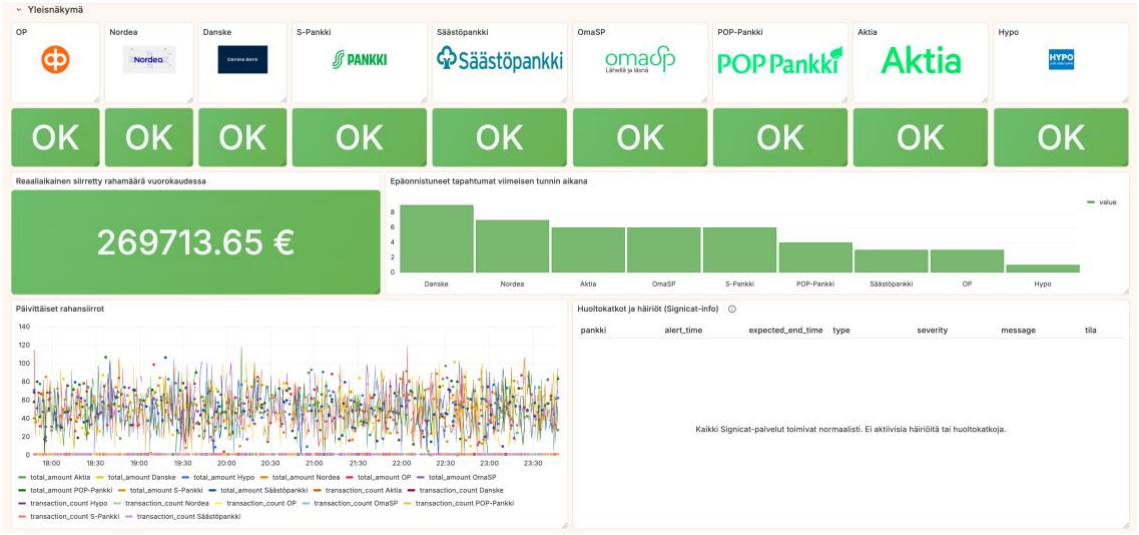
4.5 Tulosten visualisointi Grafana Cloud - ympäristössä

Tulosten visualisointi aloitettiin rakentamalla yleisnäkymä, joka tarjosi käyttäjälle reaaliaikaisen kokonaiskuvan pankkien välisistä rahansiirroista. Tämä näkymä ei liittynyt koneoppimismalleihin tai tekoälyyn, vaan toimi täysin itsenäisesti ilman ennustavia algoritmeja. Tavoitteena oli tarjota nopea katsaus nykyhetken tilanteeseen, jotta mahdolliset katkokset tai poikkeamat pankkisiirroissa nousisivat heti esiin.

Yleisnäkymän näkyvin osa oli pankkikohtainen status-paneeli, jossa kullekin pankille (esim. OP, Nordea, Danske Bank, S-Pankki, OmaSP) oli oma tilalaatikkonsa. Laatikon taustaväri, kuvake ja status muuttuivat sen mukaan, oliko pankista havaittu siirtoja viimeisen 10 (suurilla pankeilla) tai 20 (pienemmillä pankeilla) minuutin aikana. Hälytysrajojen logiikka perustui TIMESTAMP_DIFF-vertailuun BigQueryn tietueista. Pankkien logot haettiin Githubiin ladatuista kuvista ja esitettiin Image panelien avulla selkeästi tunnistettavina visuaalisina elementteinä.

Yleisnäkymään tehtiin myös paneeli, jossa visualisoitiin päivän aikana kertyneet rahansiirrot kaikilta pankeilta. Tämä toteutettiin käyttämällä CURRENT_DATE("Europe/Helsinki") -logiikkaa, jolla varmistettiin, että laskenta nollautui joka päivä klo 00:00. Tavoitteena oli saada selkeä näkymä siihen, onko rahaliikenne alkanut normaalisti aamun aikana ja kuinka suuriksi summat kehittyvät päivän mittaan. Haasteena oli kuitenkin saada SQL-kysely toimimaan oikein. Alkuvaiheessa kysely näytti koko aineiston siirtojen kokonaissumman, eikä päivämääräkohtaista arvoa.

Yleisnäkymään rakennettiin myös Signicat-paneeli, jonka tarkoituksena oli näyttää mahdolliset häiriötiedotteet ja suunnitellut huoltokatkot pankkikohtaisesti. Toteutus perustui Signicatin testirajapintaan, koska tuotantotason pääsy olisi edellyttänyt erillistä lupaa ja mahdollisesti maksullista sopimusta. Paneelissa esitettiin JSON-vastauksen perusteella pankin nimi, huoltokatkon ajankohta ja mahdollinen vaikutus. Toteutus jäi prototyyppitasolle, mutta osoitti hyvin, miten kolmannen osapuolen ilmoituksia voidaan tuoda osaksi valvontaratkaisua.



KUVA 7. Grafanan dashboardin yleisnäkymä pankkivalvontaan liittyvistä tunnusluvuista. Näkymässä korostuvat pankkikohtaiset häiriötilanteet, reaaliaikainen siirtomäärä ja epäonnistuneet tapahtumat.

Varsinaiset koneoppimismallit rakennettiin erillisissä vaiheissa, ja ne visualisoitiin omissa Grafana-riveissään. Aluksi oli tarkoitus rakentaa mallivalinta valikon kautta, mutta ml_view-muuttujaan perustuva näkymänvaihto osoittautui haasteelliseksi. Epäselväksi jäi, oliko tämänkaltaista toteutusta edes mahdollista rakentaa täysin halutulla tavalla Grafanassa. Tämän vuoksi päädyttiin lopulta ratkaisuun, jossa jokainen malli esitettiin omassa rivissään ja näkymät järjestettiin staattisesti – selkeästi ja helposti hallittavasti.

Vaikka kaikki mallit saivat tuloksensa BigQuerystä, niiden visualisointi Grafanassa toteutettiin eri tavoin rakenteellisesti ja visuaalisesti, jotta kullakin mallilla olisi sille sopiva ja käyttötilanteeseen optimoitu esitystapa. Mallien esitystapaa ohjasi ensisijaisesti se, millainen käyttöliittymäkokemus haluttiin tarjota loppukäyttäjälle: oliko tarkoitus havainnoida poikkeamia, vertailla ennusteita vai tunnistaa luokitustuloksia.

Esimerkiksi poikkeamien tunnistamiseen keskittyville malleille, kuten Z-score ja Isolation Forest, rakennettiin näkymät, joissa käytettiin Stat-paneeleita, aikasarjoja ja taulukoita. Näiden avulla käyttäjä pystyi nopeasti havaitsemaan poikkeamien määrän ja ajallisen jakautumisen sekä tarkastelemaan yksittäisiä tapauksia suoraan näkymästä.

Vertex AI Forecast -mallin kohdalla visuaalinen painopiste siirtyi rinnakkaisten aikasarjojen esittämiseen. Toteutuneet arvot ja ennusteet visualisoitiin vierekkäin samassa graafissa, mikä mahdollisti intuitiivisen vertailun mallin tarkkuudesta. Tämä edellytti tarkkaa akselien ja arvosarjojen hallintaa, jotta eri tietolähteiden datat asettuivat oikein toisiinsa nähden.

Luokittelumallin näkymä puolestaan rakentui taulukkopohjaisen tarkastelun varaan. Visualisoinnissa painottui värikoodaus ja hierarkia, joiden avulla käyttäjä pystyi tunnistamaan merkittävät tapahtumat nopeasti – esimerkiksi punaisella korostetut korkean riskin tapahtumat nousivat heti esiin listauksessa.

Vaikka näkymät erosivat toisistaan mallikohtaisesti, koko dashboardin suunnittelussa pyrittiin johdonmukaiseen visuaaliseen yhdenmukaisuuteen. Tämä tarkoitti mm. seuraavaa:

- Värit: vihreä = normaali, oranssi = tarkkailtava, punainen = poikkeama
- Fonttikoko ja rytmi: suurimmat arvot korostettiin Stat-paneeleissa, pienemmät metatiedot esitettiin yhtenäisellä tyylillä
- Numeromuotoilu: eurot ilman lyhenteitä ("4 000 €" ei "4k")
- Aikavyöhykkeet: kaikki näkymät standardoitiin muotoon Europe/Helsinki

Paneelien asettelu noudatti vasemmalta oikealle ja ylhäältä alas etenevää rakennetta. Esimerkiksi mallien kohdalla yläosassa esitettiin mallin tunnistetiedot ja arviointimittarit, ja niiden alapuolella mallin tulokset kuten poikkeamat, tapahtumalistat ja graafit.

Yhdenmukainen visuaalinen toteutus ei ollut vain esteettinen ratkaisu, vaan olennainen osa järjestelmän käytettävyyttä, luotettavuutta ja opittavuutta. Kun jokainen näkymä noudatti samoja periaatteita, käyttäjän ei tarvinnut opetella järjestelmää uudelleen mallista toiseen siirtyessään. Käyttäjä pystyi keskittymään vain olennaiseen: tiedon tulkintaan ja toimintaan sen pohjalta.

5 TESTAUS JA TULOKSET

5.1 Mallikohtaiset arviointi metodit (Precision, Recall, MAE, RMSE)

Mallien arviointi perustuu valittuihin mittareihin, jotka kuvaavat sekä mallien tarkkuutta että käytännön käyttökelpoisuutta valvontajärjestelmässä. Koska käytössä oli useita eri tyyppisiä koneoppimismalleja (poikkeamien tunnistus, ennustaminen ja luokittelu), myös arviointimetodit vaihtelivat mallityypin mukaan. Arviointi ei perustunut pelkkään laskennalliseen tarkkuuteen, vaan huomioon otettiin myös mallin rooli käytännön järjestelmässä – esimerkiksi kuinka helposti tulokset ovat tulkittavissa operatiivisessa näkymässä.

Luokittelumallille, joka toteutettiin Pythonilla, käytettiin keskeisinä arviointimittareina Precisionia ja Recallia. Vastaavasti myös poikkeavuuksien tunnistuksessa (esim. Isolation Forest) tarkasteltiin samoja mittareita soveltuvin osin.

Precision (tarkkuus) kuvaa, kuinka suuri osa mallin poikkeamiksi luokittelemista tapahtumista oli todellisia poikkeamia. Recall (tunnetaan myös herkkyytenä) kertoo, kuinka suuri osa todellisista poikkeamista malli onnistui havaitsemaan.

Näiden tasapainoa seurattiin myös F1-scoren avulla, vaikka sitä ei esitetty lopullisessa Grafana-näkymässä. Käytännössä kaikki mittarit laskettiin vertaamalla mallin tuottamia luokituksia jälkikäteen tunnistettuihin todellisiin ongelmatilanteisiin. Näin mallin liiketoimintahyöty saatiin esiin selkeämmin kuin pelkällä perinteisellä tarkkuusmittarilla (accuracy).

Aikasarjaennusteen (Vertex AI Forecast) arviointiin käytettiin puolestaan virhemittareita MAE (Mean Absolute Error) ja RMSE (Root Mean Squared Error). MAE mittaa keskimääräistä poikkeamaa ennusteen ja toteutuneen välillä, ja on tulkinnaltaan suoraviivainen: mitä pienempi MAE, sitä lähempänä malli on ollut todellisuutta. RMSE korostaa suurempia virheitä neliöittämisen ansiosta, joten se auttaa tunnistamaan mallin herkkyyden yksittäisiin suuriin heittoihin. Näillä

mittareilla saatiin hyvä kokonaiskuva siitä, kuinka ennusteet käyttäytyivät eri pankkien kohdalla, ja missä tilanteissa malli toimi parhaiten.

Jokainen mittari laskettiin suoraan BigQueryyn tallennetuista mallituloksista. Tämä mahdollisti sen, että kaikki arviointiarvot voitiin esittää Grafana-paneeleissa reaaliaikaisesti tai päivittyen automaattisesti mallin ajon jälkeen. Näin mallin laatu ei ollut pelkästään dokumentaatiotasolla, vaan sen suorituskyky oli jatkuvasti nähtävissä käyttöliittymässä.

Vaikka arviointiin käytettiin vain neljää keskeistä mittaria (Precision, Recall, MAE, RMSE), on olemassa useita muitakin arviointimetodeja. Luokittelumalleissa voitaisiin käyttää esimerkiksi Accuracy, joka kertoo kaikkien oikein luokiteltujen osuutta, tai ROC-AUC, joka mittaa mallin kykyä erotella luokat eri kynnysarvoilla. Myös Log loss ja PR-AUC ovat hyödyllisiä erityisesti todennäköisyyspohjaisille malleille ja epätasapainoiselle datalle (Géron, 2023)

Regressiomalleissa yleisiä vaihtoehtoja ovat MAPE (Mean Absolute Percentage Error), joka mittaa suhteellista virhettä, sekä R^2 (selityskerroin), joka kuvaa mallin kykyä selittää vaihtelua datassa. Aikasarjamalleille on tarjolla esimerkiksi sMAPE ja MASE, jotka ovat erityisen käyttökelpoisia silloin, kun halutaan vertailla mallia yksinkertaisiin ennustemalleihin tai välttää MAPE:n herkkyyttä nollan lähellä oleville arvoille.

Tässä työssä valitut mittarit valittiin käytännön tulkittavuuden ja analysoitavuuden perusteella. Precision ja Recall tarjoavat luokittelumalleille relevantin kuvan suorituskyvystä varsinkin poikkeamien havainnoinnissa, kun taas MAE ja RMSE tarjoavat helposti tulkittavan näkymän aikasarjaennusteiden virhemarginaaleihin. Valinnoissa painottui siis se, kuinka hyvin mittarit tukevat valvontaratkaisun tavoitteita ja kuinka suoraan ne on mahdollista liittää Grafanan visualisointeihin.

5.1.1 Z-Score

Z-score-malli otettiin käyttöön työn alkuvaiheessa yhtenä yksinkertaisimmista poikkeamien tunnistusmenetelmistä. Mallin ajatuksena oli laskea jokaiselle

pankille ja kellonajalle odotusarvo sekä keskihajonta aiempien havaintojen perusteella. Tämän jälkeen laskettiin kunkin uuden havaintopisteen Z-score, joka kuvaa, kuinka monen keskihajonnan päässä se on odotusarvosta. Jos Z-score ylitti valitun kynnysarvon (esimerkiksi 2.5 tai 3.0), havainto tulkittiin poikkeamaksi.

Mallin kehityksessä edettiin kolmessa versiossa:

- Versio 1.0.0: Ensimmäinen tuotantoversio. Toteutti peruslogiikan ilman segmentointia kellonaikojen mukaan.
- Versio 1.0.1 (päivitetty 10.3.2025): Paransi ennustetarkkuutta lisäämällä vuorokaudenajan mukaan tapahtuvan ryhmittelyn (esimerkiksi aamu, päivä, ilta).
- Versio 1.0.2 (viimeisin päivitys 31.3.2025 klo 18:03): Lisäsi pankkikohtaisen herkkyysparametrin, jonka avulla mallin reagointia voitiin hienosäätää erityisesti pienten pankkien kohdalla.

Testausvaiheessa Z-score-malli osoittautui käyttökelpoiseksi erityisesti suurten pankkien rahavirtojen seurannassa, sillä niissä havaintotiheys oli riittävän korkea laskennan luotettavuuden kannalta. Esimerkiksi OP- ja Nordea-pankkien aamusiirrot näyttivät toistuvasti samanlaisia kuvioita arkipäivisin, jolloin malli kykeni havaitsemaan poikkeavat tilanteet nopeasti. Pienempien pankkien kohdalla vaihtelu oli kuitenkin suurempaa, mikä heikensi Z-scoren toimintaa. Erityisesti tämä korostui tapauksissa, joissa siirtoja oli vähän tai niissä esiintyi epäsäännöllisiä piirteitä.

Mallin arviointi tehtiin vertaamalla sen havaitsemia poikkeamia jälkikäteen tunnistettuihin oikeisiin ongelmatilanteisiin. Precision-arvot nousivat korkeiksi suurilla pankeilla (noin 0.85), kun taas Recall vaihteli enemmän riippuen poikkeamien harvinaisuudesta. Kokonaisuutena Z-score osoittautui käyttökelpoiseksi varhaisvaiheen hälytysmallina, mutta ei riittänyt yksinään kattavaan poikkeamien tunnistukseen.

Virhemittareiden osalta mallille mitattiin seuraavat arvot viimeisimmän version (1.0.2) perusteella:

- MAE (Mean Absolute Error): 16,0
- RMSE (Root Mean Squared Error): 20,0

Nämä arvot kertovat, että keskimääräinen ennustevirhe oli noin 16 euroa per havainto, ja yksittäiset suuremmat poikkeamat nostivat RMSE-arvoa selvästi MAE:n yläpuolelle. Tämä viittaa siihen, että mallilla oli hyvät valmiudet reagoida keskimääräisiin vaihteluihin, mutta se oli herkkä yksittäisille suurille poikkeamille – mikä olikin tavoiteltua hälytyskäytön kannalta.

Mallin tulokset tallennettiin BigQuery-tauluun `anomaly_zscore_results`, josta ne haettiin Grafanaan Stat-paneeleihin ja aikasarjakuvioihin. Käyttäjälle mallin vaikutus näkyi punaisina tai oransseina korostuksina pankkikohtaisissa näkymissä, ja kokonaismäärät esitettiin erillisessä poikkeamien yhteenvetopaneelissa. Mallin yksinkertaisuus mahdollisti sen ajamisen useita kertoja päivässä ilman merkittävää laskentatehon tarvetta.

Z-score-malli toimi hyvin lähtökohtana ja vertailupisteenä muille malleille, sillä sen rakenne oli läpinäkyvä ja tulokset helposti tulkittavissa myös ilman syvällistä koneoppimisosaamista. Tämä teki siitä myös erinomaisen työkalun järjestelmän ensivaiheen validointiin ja visuaalisten hälytyspaneelien testaukseen.

5.1.2 Isolation Forest

Isolation Forest -malli rakennettiin osaksi poikkeamien tunnistamiseen keskittyvää valvontaratkaisua tilanteisiin, joissa rahansiirroissa syntyy äkillisiä ja ennakoimattomia poikkeamia. Menetelmä on erityisen hyvin soveltuva tämänkaltaiseen tehtävään, sillä se ei vaadi ennakkoon merkittyä koulutusdataa eikä oletuksia poikkeamien luonteesta. Isolation Forest toimii etsimällä datapisteitä, jotka erottuvat muista siten, että ne voidaan eristää huomattavasti vähemmillä jakovaiheilla. Tämä tekee siitä kevyen, skaalautuvan ja tehokkaan algoritmin, erityisesti silloin, kun datassa ei ole selkeää sääntömäisyyttä.

Kehitystyön aikana mallia parannettiin muutamaan otteeseen, ja lopullinen versio, V 1.0.2, otettiin käyttöön 1.2.2025 klo 10:45. Mallin ajaminen tapahtui Google Cloudin Vertex AI Workbench -ympäristössä Pythonilla, ja se perustui BigQuerystä haettuun, esikäsiteltyyn siirtodataan. Lopputulokset tallennettiin erilliseen BigQuery-tauluun, josta ne oli helppo nostaa Grafanaan valvontaa varten.

Mallin toimivuutta arvioitiin kahdella keskeisellä mittarilla: Precision ja Recall. Näiden valinta perustui siihen, että valvontaratkaisussa on tärkeää havaita mahdollisimman suuri osuus aidoista ongelmista (Recall), mutta samalla välttää liiallisia vääriä hälytyksiä (Precision), jotka kuormittaisivat käyttäjää turhaan.

Arviointitulokset olivat erittäin lupaavia:

- Precision: 91,4 %
- Recall: 89,8 %

Korkea Precision osoittaa, että suurin osa mallin ilmoittamista poikkeamista oli oikeutettuja. Toisin sanoen, malli ei tarjonnut turhia hälytyksiä, vaan sen tuottamaan dataan saattoi pääsääntöisesti luottaa. Recall-lukema puolestaan kertoo, että lähes kaikki todelliset poikkeamat havaittiin, mikä on kriittistä tilanteissa, joissa reagointiviive voi aiheuttaa merkittäviä liiketoiminnallisia seurauksia.

Näiden tulosten valossa Isolation Forest -malli osoitti arvonsa käytännön tuotantoympäristössä. Se ei ollut vain matemaattisesti toimiva, vaan myös käyttökelpoinen osa laajempaa valvontajärjestelmää. Mallin visuaalinen esitys Grafanassa rakentui paneeleista, jotka esittivät sekä kokonaiskuvan löydetyistä poikkeamista että yksittäisten tapausten tarkastelun mahdollistavat taulukot. Käyttöliittymässä käyttäjä pystyi seuraamaan mallin suorituskykyä reaaliaikaisesti, ja tarkastelemaan, milloin ja minkä pankin kohdalla poikkeamat olivat syntyneet.

Kaiken kaikkiaan Isolation Forest -mallin käyttöönotto toi järjestelmään huomattavaa lisäarvoa. Sen avulla pystyttiin havaitsemaan rahansiirroissa esiintyviä poikkeamia nopeasti ja luotettavasti, ja mallin korkea suorituskyky varmensi sen, että mallista saatavat hälytykset olivat operatiivisesti merkityksellisiä. Tämä teki siitä paitsi teknisesti kiinnostavan myös käytännössä hyödyllisen työkalun rahansiirtoprosessien valvontaan.

5.1.3 Vertex AI Forecast

Vertex AI Forecast -malli otettiin käyttöön työn aikasarjapohjaiseksi ennustemalliksi, jonka tavoitteena oli arvioida pankkien rahansiirtojen päivittäistä kertymää. Malli rakennettiin Google Cloudin Vertex AI Forecasting -palvelun avulla, joka tarjosi valmiin, skaalautuvan ja automatisoidun ympäristön ennustemallien kehitykseen. Tärkeänä perusteluna Vertexin käytölle oli kyky integroitua saumattomasti BigQuery-datapohjaan sekä mahdollisuus hyödyntää Google Cloudin hallinnoimaa infrastruktuuria ilman oman ajoympäristön ylläpitoa.

Mallin kehitys sisälsi useita vaiheita, kuten simuloidun datan muokkaamisen Vertexin vaatimaan muotoon (ml-ready CSV, sisältäen kohteen, aikaleiman ja ominaisuuksia) sekä mallin koulutuksen hallitun käyttöliittymän kautta. AutoML-vaihtoehdon sijaan valittiin "custom training" -lähestymistapa, jotta mallin rakennetta ja konekapasiteetin käyttöä voitiin hallita tarkemmin. Koulutuksessa käytettiin standard-luokan koneita, ja budjetin rajoitteet asettivat koulutuksen maksimiksi 10 node-tuntia. Tämä vastasi noin 30–50 euron kustannusta per mallin päivitys.

Lopullinen tuotantoversio, versio 1.0.3, valmistui ja ajettiin Vertex-ympäristössä 30.4.2025 klo 09:40. Mallin tulokset tallennettiin BigQuery-tauluun `rahavalvonta.predictions`, josta ne saatiin suoraan käyttöön Grafana-paneeleissa.

Mallin arviointiin käytettiin kahta keskeistä virhemittaria:

- MAE (Mean Absolute Error): 1 284,6

- RMSE (Root Mean Squared Error): 2 033,9

MAE-luku osoittaa, että mallin ennusteet poikkesivat keskimäärin noin 1 285 eurolla todellisista siirtosummista. Tämä on kohtalainen tulos, ottaen huomioon aineiston luonteen ja vaihtelut erityisesti korkeiden volyymipiikkien osalta. RMSE:n korkeampi arvo viittaa siihen, että mallilla oli vaikeuksia erityisesti yksittäisten suurten poikkeamien kanssa, jotka saattoivat liittyä suuriin kertasiirtoihin tai epätyypillisiin käyttäytymismalleihin.

Grafanassa Vertex-malli esitettiin aikasarjavisualisointina, jossa toteutuneet ja ennustetut summat näkyivät rinnakkain. Tämä mahdollisti selkeän visuaalisen vertailun ja auttoi käyttäjää havaitsemaan tilanteet, joissa malli epäonnistui tai onnistui hyvin. Useimmiten malli seurasi yleistä trendiä kohtuullisen tarkasti, mutta yksittäisinä päivinä esiintyi merkittäviä virhepiikkejä.

Kokonaisuutena Vertex AI Forecast -malli toimi teknisesti luotettavasti ja tarjosi käyttökelpoisen näkymän pankkikohtaisten rahavirtojen kehitykseen. On kuitenkin huomioitava, että malli ei tuottanut kovin tarkkoja ennusteita erityisesti pienemmillä pankkikohtaisilla aikasarjoilla tai poikkeuksellisten tapahtumien kohdalla. Tämä oli osittain odotettua, sillä opinnäytetyön rajoitettu budjetti vaikutti mallin koulutusresursseihin ja päivitystiheyteen. Koulutuksen hinta (30–50 € per ajo) esti toistuvat parannusyritykset, mutta malli toimi silti luotettavana pohjana jatkokehitykselle.

Vertex AI Forecast -mallia voidaan parantaa jatkossa esimerkiksi lisäämällä datan määrää, käyttämällä pidempää koulutusaikaa tai hyödyntämällä kehittyneempiä mallikonfiguraatioita. Jo tässä muodossaan se kuitenkin tarjosi konkreettisen esimerkin siitä, miten automatisoitu pilvipohjainen ennustemalli voidaan yhdistää visuaaliseen valvontaratkaisuun tehokkaasti ja skaalautuvasti.

5.1.4 Luokittelumalli

Luokittelumallin kehittäminen tarjosi projektiin uudenlaisen lähestymistavan rahansiirtojen valvontaan, sillä sen tavoitteena ei ollut ennustaa summia tai

havaita poikkeamia tilastollisin menetelmin, vaan nimenomaan tehdä eksplisiittinen päätös siitä, onko yksittäinen tapahtuma normaali vai potentiaalisesti poikkeava. Tämä teki mallista erittäin käytännönläheisen työkalun tilanteisiin, joissa tarvitaan selkeitä ja toimintaa ohjaavia signaaleja.

Mallin kehitys päätettiin toteuttaa Python-ympäristössä, hyödyntäen Vertex AI Workbenchin tarjoamaa skaalautuvaa infrastruktuuria sekä suoraa integraatiota BigQueryyn. Tämä mahdollisti mallin koulutuksen pilvessä ilman tarvetta siirtää tai replikoida dataa erillisiin ympäristöihin. Toisin kuin Vertex AI Forecast -mallissa, tässä ei käytetty valmista Vertex-mallinnusalustaa, vaan haluttiin rakentaa malli manuaalisesti, jotta mallin rakenne, ominaisuusvalinnat ja hyperparametrit pysyivät täysin hallinnassa. Tavoitteena oli myös testata käytännössä, kuinka hyvin käsin rakennettu malli pärjää Googlen automatisoitujen ratkaisujen rinnalla.

Valmis tuotantoversio luokittelumallista oli:

- Versio: V 1.0.4
- Päivitetty: 23.5.2025 klo 10:05

Mallin suorituskykyä mitattiin kahdella klassisella luokittelumittarilla: Precision ja Recall. Nämä mittarit olivat erityisen sopivia tähän käyttötilanteeseen, sillä ne kuvaavat mallin kykyä tehdä luotettavia päätöksiä. Toisaalta ne auttavat vähentämään virheellisiä hälytyksiä ja toisaalta varmistamaan, että todelliset poikkeamat havaitaan.

Tulokset olivat seuraavat:

- Precision: 94,6 %
- Recall: 93,1 %

Näin korkea tarkkuus ja kattavuus kertovat mallin olevan erittäin luotettava: se ei ainoastaan löytänyt lähes kaikki oikeat poikkeamat (korkea recall), vaan teki sen ilman että tuotti suurta määrää turhia hälytyksiä (korkea precision). Tämä tasapaino on erityisen tärkeä valvontakäytössä, jossa jokainen väärä hälytys vie

käyttäjän huomiota turhaan, ja jokainen havaitsematon poikkeama voi merkitä menetettyä havaintoa tai riskiä.

Grafana-paneelien suunnittelu mukautettiin palvelemaan luokittelumallin luonnetta. Pääpaino oli selkeässä ja nopeasti tulkittavassa näkymässä: tapahtumat esitettiin taulukkona, jossa jokaiselle tapahtumalle näytettiin mallin antama luokka ja siihen liittyvä riskitaso. Taulukon rivit väritettiin automaattisesti — vihreä merkitsi normaalia tilannetta, keltainen kohonnutta riskiä ja punainen selkeää poikkeamaa. Tämä visuaalinen koodaus mahdollisti käyttäjälle tehokkaan priorisoinnin ilman tarvetta syventyä yksityiskohtiin.

Lisäksi näkymään lisättiin Stat-paneelit, jotka näyttivät mallin versionumeron, viimeisimmän ajon ajankohdan sekä suorituskykymittarit. Tällä varmistettiin, että myös käyttäjät, joilla ei ollut syvällistä koneoppimisosaamista, saattoivat seurata mallin toimintaa ja ymmärtää sen luotettavuuden keskeisten lukujen avulla.

Yhteenvetona voidaan todeta, että luokittelumalli tarjosi tehokkaan ja käyttökelpoisen välineen rahansiirtojen valvontaan. Sen tarjoama selkeä päätös normaali–poikkeama-akselilla oli omiaan tukemaan valvonnan automaatiota ja nopeaa reagointia. Mallin onnistunut visuaalinen toteutus Grafanassa puolestaan varmisti, että tulokset olivat ymmärrettäviä ja toimintakelpoisia myös niille käyttäjille, joiden tehtävänä oli toimia mallin havaintojen perusteella, ei tulkita niitä tilastollisesti.

5.2 Mallien tuottama lisäarvo

Mallien käyttöönoton keskeinen tavoite ei ollut pelkästään tekninen kokeilu, vaan nimenomaan tuottaa konkreettista lisäarvoa rahansiirtojen valvontaan. Jokainen malli – Z-score, Isolation Forest, Vertex AI Forecast ja luokittelu – toi omanlaisensa näkökulman datan tulkintaan, ja yhdessä ne muodostivat monikerroksisen valvontaratkaisun, joka ylitti perinteisen järjestelmävalvonnan kyvykkyyden.

Z-score ja Isolation Forest mallit keskittyivät poikkeamien havaitsemiseen, mutta eri tavoin. Z-score tarjosi yksinkertaisen ja helposti tulkittavan menetelmän tilastollisesti epänormaalien tilanteiden tunnistamiseen. Se oli erityisen käyttökelpoinen silloin, kun haettiin nopeasti havaittavia muutoksia totutussa rahansiirtokäyttäytymisessä. Isolation Forest puolestaan kykeni löytämään monimutkaisempia poikkeamia useiden attribuuttien yhdistelmistä ja oli siten tehokkaampi löytämään hienovaraisempia häiriöitä, joita pelkkä keskiarvosta poikkeaminen ei paljasta.

Vertex AI Forecast -malli tuotti lisäarvoa ennakoitavuuden kautta. Sen avulla oli mahdollista arvioida jo aamupäivän aikana, onko rahansiirtojen kertyminen normaalilla tasolla vai ollaanko jäämässä jälkeen päivän normaalista rytmistä. Tämä ennakoiva näkymä tarjosi operatiiviselle valvonnalle uuden työkalun, joka voi mahdollistaa reagoimisen ongelmiin ennen kuin ne ehtivät kärjistyä näkyviksi häiriöiksi. Vaikka mallin tarkkuus ei ollut täydellinen, sen kyky osoittaa päivittäisiä kehitystrendejä oli selkeä askel perinteistä reaaliaikaista valvontaa pidemmälle.

luokittelumalli puolestaan antoi selkeän ja yksinkertaisen signaalin siitä, milloin yksittäinen tapahtuma vaikutti poikkeavalta. Sen suurin lisäarvo oli päätöksenteon tukemisessa: mallin tuottama luokitus mahdollisti automatisoidun priorisoinnin ja riskiluokituksen ilman tarvetta manuaaliselle seulonnalle. Käyttäjän näkökulmasta tämä vähensi virheellisiin päätelmiin johtavia arviointivirheitä ja säästi aikaa.

Kokonaisuutena mallien suurin lisäarvo oli kuitenkin siinä, että ne yhdessä muodostivat toisiaan täydentävän analyysikerroksen, ei pelkästään rikastaen näkymää, vaan nostaen sen uudelle tasolle. Yksi malli ei välttämättä pystynyt yksin tuottamaan täydellistä valvontakuvaa, mutta yhdistettynä ne pystyivät havaitsemaan sekä yksittäisiä virheitä että laajempia poikkeamia, ennakoimaan tulevaa ja tukemaan päätöksentekoa.

Toinen merkittävä hyöty oli mallien visuaalinen läpinäkyvyys Grafanassa. Tulokset eivät jääneet pelkästään taustalla pyöriviksi algoritmeiksi, vaan ne tuotiin käyttöliittymän kautta selkeiksi ja toimintakelpoisiksi näkymiksi. Tämä

mahdollisti myös ei-teknisten käyttäjien osallistumisen valvontaan ja analyysiin, mikä edelleen lisäsi järjestelmän käytettävyyttä ja vaikuttavuutta.

Mallien rakentaminen, arviointi ja käyttöönotto veivät resursseja, mutta lopputuloksena saavutettiin järjestelmä, joka tarjoaa monipuolisempaa ja älykkäämpää rahavirtojen valvontaa kuin perinteiset manuaaliset tai vain reaaliaikaan perustuvat ratkaisut. Tämän vuoksi voidaan perustellusti todeta, että koneoppimismallit toivat merkittävää lisäarvoa niin datan hyödyntämiseen kuin järjestelmän kokonaistoiminnallisuuteen.

5.3 Poikkeamien tunnistuksen tarkkuus (Oma malli vs Vertex AI:n)

Yksi keskeisistä tavoitteista tämän opinnäytetyön mallikehityksessä oli arvioida, kuinka hyvin omalla Python-pohjaisella toteutuksella rakennettavat koneoppimismallit pärjäävät verrattuna Google Cloudin tarjoamaan automatisoituun Vertex AI Forecast -palveluun. Vaikka mallien ensisijainen käyttötarkoitus ei ollut täysin identtinen – osa keskittyi poikkeamien tunnistamiseen ja osa ennustamiseen – niiden tuloksia voidaan vertailla yleisellä tasolla tarkkuuden ja käytettävyyden näkökulmasta.

Omat mallit (Z-score ja Isolation Forest) keskittyivät nimenomaan poikkeamien tunnistamiseen. Näiden mallien kohdalla pystyttiin mittaamaan suoraan Precision- ja Recall-arvoja, joilla arvioitiin, kuinka tarkasti mallit löysivät todellisia poikkeamia ja kuinka kattavasti ne tunnistivat kaikki poikkeavat tapaukset. Esimerkiksi Isolation Forest -malli saavutti Precision-tuloksen 91,4 % ja Recall-arvon 89,8 %, mikä viittaa siihen, että suurin osa havaituista poikkeamista oli todellisia (vähän vääriä hälytyksiä) ja myös suurin osa todellisista poikkeamista löydettiin (vähän puuttuvia havaintoja).

Vertex AI Forecast puolestaan toimii enemmän ennustemallina kuin poikkeamaluokittelijana. Tästä syystä sen arviointi ei tapahdu perinteisillä Precision- tai Recall-mittareilla, vaan sillä käytettiin MAE- ja RMSE-mittareita, jotka kuvaavat ennusteiden ja toteutuneiden arvojen välistä eroa. Vaikka ennustemalli ei suoranaisesti tuottanut "poikkeamia", sen ennustearvoista saattoi

tulkita, milloin todellinen kehitys poikkesi merkittävästi mallin arvioimasta linjasta — toisin sanoen: ennusteen ja toteuman ero toimi epäsuorana poikkeamasignaalina.

Kun näitä kahta lähestymistapaa vertaillaan, voidaan havaita seuraavia eroja:

- Tulkittavuus ja välittömyys: Pythonilla toteutetut poikkeamamallit tuottivat suoraan binary-luokituksen (esim. poikkeama kyllä/ei), mikä teki niistä helpommin operatiiviseen käyttöön soveltuvia. Käyttäjä sai välittömästi tiedon mahdollisesta ongelmatilanteesta.
- Monipuolisuus ja mukautuvuus: Vertex AI Forecast puolestaan tarjosi jatkuvan ennusteen kehityksestä, mikä oli hyödyllistä esimerkiksi varhaisvaroitusjärjestelmänä. Sen käyttö oli kuitenkin enemmän tulkinnanvaraista — vaati ihmisen tekemän raja-arvon, milloin ennusteen ja toteuman ero on ”tarpeeksi suuri” ollakseen merkittävä.

- Tekninen hallittavuus: Omien mallien kohdalla oli enemmän vapautta säätää mallien rakennetta, hyperparametreja ja dataa. Vertex AI:n automatisoitu luonne vähensi säätömahdollisuuksia, mutta nopeutti käyttöönottoa ja tarjosi valmiin, skaalautuvan ympäristön ilman infrastruktuurivastuuta.

Yhteenvetona voidaan todeta, että poikkeamien tunnistuksessa Python-pohjaiset mallit osoittautuivat tarkemmiksi ja suoremmin operatiiviseen valvontaan soveltuviksi. Vertex AI Forecast toimi erinomaisesti yleistrendin arvioimisessa ja mahdollisti pidemmän aikavälin ennakoinnin, mutta ei yksinään ollut riittävä työkalu kriittisten yksittäisten poikkeamien nopeaan tunnistamiseen.

Tämän vertailun perusteella voidaan perustellusti todeta, että paras ratkaisu ei ollut joko–tai, vaan sekä–että: omat mallit toimivat nopeina poikkeamanpaljastajina, kun taas Vertexin ennusteet loivat kontekstin päivän normaalille kehitykselle. Näiden yhdistäminen samaan visuaaliseen käyttöliittymään toi järjestelmään sekä syvyyttä että reagointikykyä, ja siten lisäarvoa valvontatyöhön.

5.4 Vasteaika, suorituskyky ja kustannusseuranta

Vasteajan, suorituskyvyn ja kustannusten seuranta oli olennainen osa projektin kokonaisarviointia, sillä järjestelmä rakentui Google Cloud Platformin palveluiden sekä Grafana Cloudin varaan. Nämä tekijät vaikuttivat suoraan ratkaisun käytettävyyteen ja jatkokehityspotentiaaliin.

Vasteaika pysyi suurimmaksi osaksi riittävän nopeana: Stat- ja Table-paneelit päivittyivät 1–5 minuutin välein, ja tärkeimmät kyselyt palauttivat tiedot tyypillisesti muutamassa sekunnissa. Kuitenkin Grafanan datan latauksessa havaittiin, että etenkin suuria aikasarjoja tai useita pankkeja kattavissa näkymissä tiedonhaku saattoi kestää useita sekunteja. Tämä viive korostui tilanteissa, joissa BigQueryn kyselyt kohdistuivat laajoihin simuloituihin datamassoihin. Viiveet voivat osin johtua myös tekijän kokemattomuudesta Grafanan optimoinnissa, mutta se osoitti, että jatkokehityksessä datan määrän kasvuun on syytä varautua huolellisella suunnittelulla ja suorituskykytestauksella.

Suorituskyvyn osalta BigQuery suoriutui hyvin isojen tietomäärien käsittelystä, ja Grafanan liitännät toimivat vakaasti ilman katkoksia. Kyselyjen rakenteessa ja ajastusväleissä onnistuttiin löytämään tasapaino, joka mahdollisti sekä ajantasaisen näkymän että maltillisen kuormituksen ilman, että käyttäjä koki järjestelmän hitaaksi.

Kustannusseuranta oli kriittinen erityisesti Vertex AI Forecast -mallin kohdalla. Alkuvaiheessa näytti siltä, että kustannukset voisivat nousta korkeiksi, erityisesti Vertexin laskentatehon ja BigQueryn kyselykulujen yhdistelmän vuoksi. Lopulta budjetti pysyi kuitenkin hyvin hallinnassa: Vertexin koulutukselle asetettiin rajoitteeksi 10 node-tuntia, ja BigQueryn laskutus pysyi kohtuullisella tasolla, koska data oli simuloitua ja kyselyt pystyttiin ajamaan suunnitellusti.

Yhteenvetona voidaan todeta, että järjestelmä toimi vasteajan ja suorituskyvyn osalta riittävästi prototyyppivaiheessa, ja kustannusten osalta hanke saatiin päätökseen yllättävän kustannustehokkaasti. Samalla se toi esiin kehityskohteita,

jotka tulisi huomioida erityisesti, jos järjestelmä vietäisiin suuremmassa mittakaavassa tuotantoon.

5.5 Kehityksen aikana esiin nousseet haasteet ja ratkaisut

Projektin edetessä nousi esiin useita teknisiä ja rakenteellisia haasteita, jotka vaativat jatkuvaa ratkaisukeskeistä otetta. Osa ongelmista oli teknisiä ja liittyivät järjestelmäintegraatioihin, mutta merkittävä osa heijasteli myös suunnitteluvaiheen puutteita ja projektinhallinnallisia seikkoja.

Jo alkuvaiheessa kävi ilmi, että projektin kokonaiskuva ja tavoiteasetanta eivät olleet täysin selkeitä. Eri komponentit, kuten data, koneoppimismallit ja Grafanan visualisointi, kyllä tunnistettiin, mutta niiden välinen yhteys ja prioriteettijärjestys muotoutuivat vasta työn edetessä. Tämä vaikeutti työn rajaamista ja vaiheistusta ja johti siihen, että osa toiminnoista jouduttiin tekemään uudelleen, kun järjestelmän arkkitehtuuri ja käyttötarkoitus tarkentuivat.

Suunnittelun puutteet näkyivät erityisesti siinä, ettei aluksi tehty konkreettista toimintasuunnitelmaa mallien kehittämiselle, versionhallinnalle ja visualisoinnille. Tämä aiheutti turhaa iterointia ja teki työn alkuvaiheesta tehottoman. Myöhemmin työn rajaukseksi muotoiltiin selkeämpi painotus: opinnäytetyö keskittyy koneoppimiseen perustuvan valvontaratkaisun rakentamiseen Google Cloud -alustalla hyödyntäen vain rajattua joukkoa palveluita. GCP:n mahdollisuuksia ei hyödynnetty laajasti, esimerkiksi Vertex AI:n lisäominaisuudet, Pub/Sub-viestijonot, automatisoidut pipeline-ratkaisut tai useat vaihtoehtoiset tietolähteet jäivät käyttämättä. Samoin Grafanassa käytettiin vain BigQueryä datalähteenä, vaikka teknisesti olisi ollut mahdollista tuoda mukaan esimerkiksi Cloud Monitoringin mittareita, Stackdriver-lokeja tai Prometheus-yhteensopivia mittauspisteitä.

Yksi keskeisistä teknisistä haasteista liittyi päiväkohtaisten summien esittämiseen Grafanassa. BigQuery-kyselyt palauttivat oletuksena koko aineiston summat, jolloin oli vaikeaa esittää päivän alusta kertyneet siirtomäärät.

Ratkaisuksi otettiin käyttöön CURRENT_DATE("Europe/Helsinki") -logiikka ja aikaleimojen tarkempi muotoilu.

Toinen huomattava haaste oli Signicatin häiriötiedotteiden visualisointi. Vaikka testirajapinta oli käytettävissä, tuotantotason käyttö vaati maksullista lisenssiä, joten toteutus jäi kokeilutasolle.

Visualisoinnin rakenteellinen suunnittelu osoittautui niin ikään ongelmalliseksi. Aluksi pyrittiin rakentamaan valintalogiikka mallien välillä käyttäen ml_view-muuttujaa, mutta teknisten rajoitteiden vuoksi tämä korvattiin erillisillä riveillä (rows) kullekin mallille, mikä lopulta osoittautui käyttäjäystävällisemmäksi ratkaisuksi.

Mallien versionhallinta ja tulosten ajantasaisuus tuottivat myös päänvaivaa, mutta ratkaistiin luomalla versionumerot ja tallentamalla tulokset BigQuery-tauluihin, joista ne luettiin Grafanaan yhdessä metatietojen kanssa.

Erityisesti Vertex AI Forecast -mallin käyttö nosti esiin muotoiluvaatimuksia, jotka poikkesivat Pythonilla rakennetuista malleista. Datan muokkaaminen Vertexin hyväksymään aikasarjaformaattiin vei aikaa ja vaati uudenlaista ymmärrystä mallin sisäisestä logiikasta. Samalla opittiin, kuinka erilaiset mallityypit asettavat omat vaatimuksensa sekä datalle että visualisoinnille.

Kokonaisuudessaan kehitystyön haasteet toivat esiin tarpeen suunnitelmallisuudelle, vaiheistukselle ja rajausten selkeyttämiselle. Moni ongelma olisi voitu välttää paremmalla alkuvaiheen määrittelyllä. Toisaalta iteratiivinen työskentelytapa toi myös joustavuutta ja mahdollisti oppimisen matkan varrella – erityisesti GCP:n, BigQueryn ja Grafanan yhdistämisessä toimivaksi koneoppimispohjaiseksi valvontaratkaisuksi.

6 POHDINTA JA ARVIOINTI

6.1 Tavoitteiden saavuttaminen ja arvio projektin onnistumisesta

Opinnäytetyön tavoitteena oli kehittää koneoppimiseen perustuva valvontaratkaisu yrityksen ja suomalaisten pankkien välisten rahansiirtojen seurantaan. Projektin tuli kattaa koko prosessi datan simuloinnista mallien rakentamiseen, tulosten visualisointiin ja järjestelmän testaamiseen käytännössä.

Tarkasteltaessa työn etenemistä ja lopputulosta voidaan todeta, että kaikki keskeiset tavoitteet saavutettiin. Ensinnäkin onnistuttiin rakentamaan tekninen prototyyppi, joka kattoi koko data-analytiikan ketjun: simuloidun datan luominen, sen esikäsittely ja lataaminen BigQueryyn, erilaisten koneoppimismallien kouluttaminen sekä tulosten esittäminen Grafana-ympäristössä. Rakennettu järjestelmä oli toimiva, ja se kykeni havaitsemaan sekä ennustamaan poikkeavuuksia siirroissa mallien avulla.

Toiseksi työ täytti vaatimukset teknologioiden hyödyntämisen osalta. Google Cloud Platform valittiin alustaksi, ja sen palvelut – erityisesti BigQuery, Vertex AI ja Workbench integroituvat hyvin kokonaisuuteen. Koneoppimismalleja onnistuttiin toteuttamaan sekä valvotun (luokittelu) että valvomattoman (Isolation Forest) oppimisen logiikalla, ja lisäksi kokeiltiin ennakoivaa aikasarjamallia (Vertex Forecast). Lisäksi perinteinen sääntöpohjainen Z-Score-menetelmä otettiin mukaan referenssimalliksi.

Kolmantena tavoitteena oli kehittää valvontanäkymä, joka olisi käytännöllinen ja helposti tulkittava. Tässä Grafana osoittautui onnistuneeksi valinnaksi. Vaikka teknisiä kompromisseja jouduttiin tekemään käyttöliittymän osalta (esimerkiksi alasvetovalikon sijaan käytettiin rivikohtaisia näkymiä), kokonaisuudesta tuli visuaalisesti selkeä, pankkikohtaisesti eritelty ja reaaliaikaisesti päivittyvä.

Ratkaisun onnistumista voidaan arvioida myös saavutettujen tulosten perusteella. Itse rakennettu luokittelumalli toimi erittäin tarkasti, ja sen

suorituskyky (Precision 94.6, Recall 93.1) osoitti koneoppimisen konkreettisen hyödyn valvonnassa. Lisäksi järjestelmä toimi vakaasti ja reagoi luotettavasti simuloituihin poikkeamiin, mikä vahvisti käytettävyyttä.

Yhteenvetona voidaan todeta, että opinnäytetyön tavoitteet saavutettiin kattavasti. Rakennettu järjestelmä toimii todellisessa valvontatilanteessa prototyyppinä, jota voidaan jatkossa kehittää eteenpäin ja laajentaa muille osa-alueille. Lisäksi työ tarjosi arvokasta kokemusta teknologioista, joita voidaan soveltaa myös muissa data-analytiikkaan ja valvontaan liittyvissä projekteissa.

6.2 Projektin tuotantovalmius, luotettavuus ja skaalautuvuus

Rakennettua järjestelmää tarkasteltiin paitsi kehitysvaiheen onnistumisen kautta, myös sen perusteella, kuinka hyvin se soveltuisi käytettäväksi yrityksen tuotantoympäristössä. Arviointi kohdistui järjestelmän vakautta, jatkuvaa käytettävyyttä, laajennettavuutta ja ylläpidettävyyttä koskeviin osa-alueisiin.

Tuotantovalmiuden osalta ratkaisu on rakenteeltaan ja teknologialtaan valmis siirrettäväksi tosielämän käyttöön. Järjestelmä koostuu modulaarisista komponenteista, jotka on rakennettu Google Cloud Platformin hallittujen palveluiden (BigQuery, Vertex AI, Workbench) varaan. Näin ollen ratkaisu on skaalautuva, helposti ylläpidettävä ja muokattavissa ilman, että yksittäiset osat häiritsevät kokonaisuutta. Mallien ajaminen, päivitys ja tulosten käsittely onnistuvat erillisinä prosesseina, mikä mahdollistaa esimerkiksi jatkuvan kehityksen ja versionhallinnan ilman järjestelmäkatkoksia.

Luotettavuus osoittautui korkeaksi koko kehitystyön ajan. Simuloidulla aineistolla suoritetut testit osoittivat, että järjestelmä pystyy toistettavasti tuottamaan relevanttia analytiikkaa ja visualisoimaan tulokset Grafanassa ilman viivettä tai teknisiä ongelmia. Mallien suorituskyky oli ennakoitavissa, ja tulokset saatiin talteen BigQueryyn loogisessa rakenteessa. Käyttöliittymä toimi vakaasti myös eri näkymien välillä navigoitaessa.

On kuitenkin tärkeä huomioida, että tuotantodatan rakenne ja sisältö voivat poiketa merkittävästi tässä työssä käytetystä simuloidusta aineistosta. Todellisessa käytössä tapahtumat voivat olla epäsäännöllisempiä, sisältää poikkeavia kausivaihteluita tai heijastaa pankkikohtaisia teknisiä erityispiirteitä, joita simuloidussa datassa ei ole täysin pystytty jäljittelemään. Tämän seurauksena mallien ennusteet eivät välttämättä ole yhtä tarkkoja tuotantoympäristössä kuin tässä kehitysvaiheessa. Siksi ennustemalleja on suositeltavaa validoida uudelleen tuotantodatan avulla ja jatkokouluttaa niitä tilanteen mukaan.

Kustannusten näkökulmasta järjestelmä rakennettiin kehitysvaiheessa kustannustehokkaasti: esimerkiksi Vertex AI Forecast -mallin koulutus maksoi noin (3x) 30-50 euroa ja Grafana Cloudin käyttö 19 euroa kuukaudessa. On kuitenkin huomioitava, että yrityksen tuotantoympäristössä kustannukset eivät ole ensisijainen rajoite. Ratkaisuja voidaan ottaa käyttöön liiketoiminnan tarpeiden mukaan ilman, että pienet kulut muodostavat esteen. Tämä antaa laajemmat mahdollisuudet hyödyntää Vertex AI:n ja muiden GCP-palveluiden täysiä ominaisuuksia skaalautuvasti.

Erityisen positiivista on se, että järjestelmä ei ole sidottu pelkästään pankkien rahansiirtoihin. Arkkitehtuuri ja tekninen toteutus ovat sellaisia, että ratkaisu on helposti sovellettavissa myös muihin valvontaprosesseihin. Esimerkiksi käyttäjäpolkujen seuranta, talletusten käsittely tai asiakastunnistuksen poikkeamien analysointi voidaan toteuttaa samalla mallipohjalla pienillä mukautuksilla. Tämä tekee ratkaisusta strategisesti arvokkaan työkalun, jolla voidaan kehittää yrityksen data-analytiikkaa laajemminkin.

6.3 Kehittäjän näkökulmat ja opit

Opinnäytetyön toteutus tarjosi ainutlaatuisen mahdollisuuden yhdistää oma kiinnostus koneoppimiseen ja järjestelmävalvontaan konkreettisessa kehitysprojektissa. Työ antoi tilaisuuden syventää teknistä osaamista koneoppimismalleista, pilvipalveluista sekä valvonnan visualisoinnista, mutta

samalla tarjosi arvokasta käytännön kokemusta siitä, miten kehitysprosessia voidaan viedä eteenpäin kokonaisuutena.

Kehitysympäristön valinta oli tärkeä osa projektin aloitusvaihetta. Vaikka yrityksellä on käytössä Google Cloud Platform (GCP) ja sen tarjoamat tuotantokelpoiset työkalut, koettiin hyödylliseksi toteuttaa ensimmäinen versio projektista täysin omassa erillisessä ympäristössä. Tähän ratkaisuun päädyttiin, koska oman ympäristön kautta oli mahdollista testata laajasti erilaisia vaihtoehtoja ilman sisäisiä hyväksymiskierroksia tai muita byrokraattisia rajoitteita. Vaikka tämä toi mukanaan joitakin teknisiä rajoitteita – erityisesti suorituskykyyn ja käytettävissä oleviin resursseihin liittyen – se antoi vapauden kokeilla, epäonnistua ja löytää paras mahdollinen toteutustapa ilman organisaation prosessien hidastavaa vaikutusta. Tästä lähtökohdasta katsottuna oma ympäristö toimi erinomaisena kehitysalustana, ja projektin tulokset voidaan nyt siirtää yrityksen testiympäristöön ja siitä edelleen tuotantoon.

Koneoppimisen saralla projektissa saatiin syvällinen käsitys siitä, kuinka erilaiset mallit toimivat, ja miten ne vaikuttavat valvonnan tuloksiin. Mallien koulutus, arviointi ja käytännön toteutus tarjosivat kokemusta sekä valvotun että valvomattoman oppimisen soveltamisesta. Oli erityisen opettavaista havaita, että yksinkertaisella rakenteella rakennettu Python-pohjainen luokittelumalli toimi merkittävästi paremmin kuin Vertex AI Forecast -malli, jonka rakentamiseen käytettiin enemmän resursseja ja jonka käyttö oli maksullista. Tämä havainnollisti, että tekninen ymmärrys ja datan hyvä tuntemus voivat tuottaa laadukkaampia tuloksia kuin automaattisesti rakennettu, mutta vähemmän kontrolloitu malli.

Projektin aikana opittiin myös käytännön taitoja pilvipohjaisten työkalujen, kuten BigQueryn, Vertex AI:n ja Workbenchin käytössä. Näiden hallinta mahdollisti mallien ajastamisen, versionhallinnan ja tulosten tehokkaan hyödyntämisen visualisointikerroksessa. Grafana osoittautui hyväksi valinnaksi, ja vaikka alasvetovalikkopohjainen näkymien hallinta ei lopulta osoittautunut teknisesti toimivaksi ratkaisuksi, korvaava toteutus rivikohtaisilla näkymillä toimi hyvin käytännössä. Ratkaisut syntyivät kokeilun ja iteratiivisen kehityksen kautta.

Lisäksi opin paljon versionhallinnan ja dokumentoinnin merkityksestä. GitHubin hyödyntäminen paitsi mallien hallintaan myös pankkilogojen ja dashboard-konfiguraatioiden säilyttämiseen osoittautui käytännössä hyödylliseksi, ja se teki projektista siirrettävän ja helposti laajennettavan.

Yhteenvetona voidaan todeta, että tämä opinnäytetyö tarjosi sekä teknistä oppia että kokemusta ketterästä järjestelmäkehityksestä. Oman ympäristön käyttö antoi vapauden oppia, kokeilla ja kehittyä. Nyt projektin tulokset voidaan siirtää vaiheittain osaksi yrityksen virallista testiympäristöä ja edelleen tuotantoon, mikäli mallien toimivuus vahvistetaan myös oikealla datalla.

6.4 Jatkokehitys ja mahdolliset laajennukset

Projektin aikana toteutettu järjestelmä osoitti, että koneoppimismalleilla voidaan tukea pankkien välisten rahansiirtojen valvontaa jopa lähes reaaliaikaisesti. Ratkaisu perustuu moderniin teknologiapinoon, jossa hyödynnettiin Google Cloudin palveluita sekä Grafanaa visualisointikerroksena. Vaikka järjestelmä toimii jo nyt käyttökelpoisena kokonaisuutena, sillä on useita konkreettisia jatkokehitysmahdollisuuksia ja laajennuspolkuja sekä teknisesti että toiminnallisesti.

Yksi selkeimmistä jatkokehitysalueista liittyy tuotantodatan käyttöön. Koska työssä käytettiin simuloitua dataa, mallit eivät ole vielä altistuneet aidolle, epätäydelliselle ja monimutkaiselle tuotantodatalle, jossa voi esiintyä esimerkiksi virhemerkintöjä, erilaisia siirtosyklejä ja pankkikohtaisia erityispiirteitä. Erityisesti pankkien rahansiirtojen ennustaminen osoittautui yllättävän vaikeaksi, koska pankit toimivat itsenäisinä yrityksinä eikä niiden aikatauluista, huoltokatkoksista tai toimintatavoista ole avoimesti saatavilla tarkkaa ja reaaliaikaista tietoa. Tämä luo ennustemalleille huomattavia haasteita, sillä jopa kaikkein kehittynein malli ei kykene ennakoimaan esimerkiksi pankin teknistä huoltokatkosta, joka vaikuttaa siirtoihin, jos tietoa siitä ei ole saatavilla. Tästä huolimatta mallinnusta tarvittiin – ei välttämättä siksi, että se antaisi täysin aukottomia ennusteita, vaan siksi, että sen avulla voidaan kehittää ja testata koneoppimispohjaisia valvontaratkaisuja tulevaisuutta varten ja soveltaa niitä muihin valvonnan osa-alueisiin.

Tulevaisuuden näkökulmasta myös käytettyjen työkalujen kehitys avaa uusia mahdollisuuksia. Tämä opinnäytetyö toteutettiin Grafana Cloudin versiolla 11.6, mutta jo julkaistu Grafana 12 tuo merkittäviä parannuksia erityisesti käyttöliittymien rakentamiseen, datasidontaan ja dynaamisiin näkymiin.

Grafana 12:ssa on muun muassa parannetut muuttujien ehdollisuudet, joiden avulla näkymiä voidaan näyttää tai piilottaa entistä tarkemmin käyttäjän valintojen perusteella (*Grafana Labs*, 2025d). Dynamic panels -ominaisuus, joka mahdollistaa sisältöjen luomisen suoraan datan perusteella ilman erillistä konfigurointia. Parempi tuki JSONata- ja DataFrames-transformaatioille, jotka mahdollistavat monimutkaisempien visualisointilogiikoiden rakentamisen suoraan Grafanan sisällä. Kehittyneempi alertointi ja alert group management, joka tekee hälytyksistä tarkempia ja helpommin hallittavia esimerkiksi pankkikohtaisissa ongelmissa. Näitä uusia ominaisuuksia voidaan hyödyntää järjestelmän tulevissa versioissa, erityisesti kun halutaan tarjota käyttäjille kohdennetumpia näkymiä ja lisätä automaattista reagointia poikkeamiin. Grafanan kehitys osoittaa myös, että valvontatyökalut itsessään muuttuvat älykkäämmiksi ja voivat jatkossa vähentää ulkoisen sovelluskoodin tarvetta.

Toinen keskeinen jatkokehityssuunta liittyy laajennettavuuteen. Rakennettu malli ei ole rajoittunut pelkästään pankkisiirtojen valvontaan, vaan sen logiikkaa ja arkkitehtuuria voidaan helposti soveltaa muuhun liiketoimintavalvontaan. Esimerkiksi asiakaskäyttäytymisen seuranta, kirjautumisvirheiden poikkeavuudet ovat kaikki alueita, joissa vastaavanlaista valvontaa voidaan toteuttaa pienin muutoksin. Koska arkkitehtuuri perustuu modulaarisiin komponentteihin, kuten BigQuery, Vertex AI ja Grafana, uusia datalähteitä tai mallityyppejä voidaan lisätä suhteellisen vähällä kehitystyöllä.

Yhteenvetona voidaan todeta, että järjestelmä toimii hyvänä pohjana koneoppimisen laajemmalle käyttöönotolle valvonnassa. Vaikka valvonnan kohde – pankkien väliset siirrot – itsessään on osittain vaikeasti ennustettava, rakenteellinen ratkaisu on joustava ja laajennuskelpoinen. Yhdistettynä kehittyviin työkaluihin, kuten Grafana 12:een, jatkokehitys voi viedä järjestelmää entistä lähemmäs automaattista ja luotettavaa päätöksenteon tukijärjestelmää

7 YHTEENVETO

7.1 Keskeiset havainnot ja tulokset

Opinnäytetyön keskeinen tavoite oli selvittää, voidaanko koneoppimismalleja hyödyntää pankkien ja yritysten välisten rahansiirtojen valvonnassa tehokkaasti ja skaalautuvasti. Projektin lopputuloksena syntyi kokonaisvaltainen valvontaratkaisu, joka yhdistää useita koneoppimismalleja, pilvipohjaisen arkkitehtuurin sekä käytännönläheisen visualisointikerroksen Grafanassa. Työn aikana toteutettiin neljä erilaista analyysimallia: Z-Score-baseline, poikkeavuuksien tunnistus (Isolation Forest), aikasarjaennuste (Vertex AI Forecast) ja luokittelumalli. Tulokset osoittivat, että koneoppiminen tarjoaa selkeää lisäarvoa valvontatyöhön erityisesti silloin, kun se yhdistetään myös perinteisiin sääntöpohjaisiin valvontamekanismeihin.

Mallikohtaisten arvioiden perusteella parhaimmin toimi itse rakennettu luokittelumalli, joka saavutti erittäin korkean tarkkuuden (Precision 94.6 / Recall 93.1). Tämä oli yllättävää erityisesti siksi, että Vertex AI Forecast -mallin toteutus oli huomattavasti kalliimpi (10 node-tunnin laskutus, (3x) n. 50 € kustannus), mutta ennustetarkkuus jäi selvästi heikommaksi (MAE 1284.6 / RMSE 2033.9). Tulokset osoittavat, että kevyt ja kohdennettu malli voi joskus olla käytännössä tehokkaampi kuin raskas, automaattinen pilvipalvelu. Tällöin myös mallin rakentajan ymmärrys datasta ja ongelmakontekstista nousee keskeiseen rooliin.

Valvonnan näkökulmasta tärkeäksi havaittiin myös se, ettei järjestelmä voinut nojata pelkästään koneoppimiseen. Koska pankit toimivat itsenäisinä toimijoina eikä niistä saada reaaliaikaista teknistä tietoa esimerkiksi huoltokatkoista tai siirtoviiveistä, ennustemallit eivät aina kykene havaitsemaan liiketoiminnallisesti merkittäviä tilanteita. Siksi Grafanaan rakennettiin myös perinteisiä sääntöpohjaisia näkymiä, kuten yksinkertaiset pankkikohtaiset siirtonäkymät sekä Signicat-palvelun tilannekuva, joka tuo pankkien häiriöt näkyville webhook-integraation kautta.

Teknisen onnistumisen lisäksi työssä nousi esiin myös kehityskohteita. Aikataulussa ei aivan pysytty, ja erityisesti työn alkuvaiheessa suunnittelu oli riittämätöntä, mikä johti joidenkin osien rönsyilyyn ja vaihtoehtoisten ratkaisujen testaamiseen useaan otteeseen. Tämä opetti projektinhallinnan näkökulmasta sen, kuinka tärkeää on määritellä selkeä rakenne ja etenemispolku jo varhaisessa vaiheessa, jotta työ pysyy hallittuna ja resurssit kohdistuvat tehokkaasti.

Toinen tärkeä havainto liittyi pilvipalveluiden kustannusrakenteeseen. Vaikka työssä hyödynnettiin Google Cloudin koekäyttötarjouksia ja ilmaisia palveluja, on syytä huomioida, että tuotantoympäristössä vastaavat ratkaisut voivat tuottaa merkittäviä kuukausikustannuksia – erityisesti jos datan määrä, säilytysaika ja mallien käyttö kasvaa. Tulevaisuudessa on tärkeää miettiä, mitä dataa säilytetään, kuinka pitkään ja missä muodossa. Kaikkea ei tarvitse pitää saatavilla reaaliajassa, ja jo pelkkä arkistoinnin suunnittelu voi tuoda huomattavia säästöjä. Pilvipalveluiden vahvuus on niiden skaalautuvuus ja joustavuus, mutta samalla niiden kustannukset voivat yllättää, ellei käyttöä seurata ja optimoida aktiivisesti.
Kaiken kaikkiaan työn keskeinen tulos on se, että koneoppimista voidaan hyödyntää osana liiketoimintakriittistä valvontaa ja että tällainen järjestelmä on teknisesti ja toiminnallisesti toteuttamiskelpoinen myös pienessä mittakaavassa. Tulokset ja opit tarjoavat vankan perustan järjestelmän viemiselle tuotantoon sekä laajentamiselle muihin valvontakohteisiin yrityksen sisällä.

7.2 Suositukset ja johtopäätökset

Tämän opinnäytetyön perusteella voidaan todeta, että koneoppimisen ja pilvipohjaisten ratkaisujen yhdistäminen rahansiirtojen valvontaan tarjoaa huomattavaa potentiaalia, mutta vaatii samalla huolellista suunnittelua, testaamista ja kustannustietoisuutta. Projektin keskeinen tavoite, eli koneoppimiseen perustuvan valvontaratkaisun rakentaminen pankkien ja yrityksen välisten siirtojen analysoimiseksi, saavutettiin pääpiirteissään. Toteutettu kokonaisuus sisälsi neljä eri analyysimallia (Z-Score, Isolation Forest,

Vertex AI Forecast sekä Pythonilla tehty luokittelumalli) ja onnistui visualisoimaan niiden tulokset selkeästi Grafana-ympäristössä.

Mallien tarkkuudessa havaittiin selkeitä eroja. Erityisesti Pythonilla toteutettu luokittelumalli saavutti korkeimmat tarkkuusluvut (Recall 93,1 %, Precision 94,6 %), kun taas Googlen Vertex AI Forecast jäi huomattavasti heikommaksi ennustetarkkuudessa (MAE 1284,6 / RMSE 2033,9). Tämä ero oli yllättävä – erityisesti kun otetaan huomioon, että Forecast-mallin koulutus maksoi sen (3x) 30-50 euroa ja käytti 10 node-tuntia laskentaresursseja. Vaikka tämä summa on kehittäjälle huomattava, yritystasolla se on varsin pieni panostus potentiaalisesti toimivasta mallista. Tästä huolimatta mallin heikko suorituskyky herättää tarpeen jatkotutkimukselle: on tärkeää selvittää, voisiko tuotantodatan avulla saavutettavissa olla huomattavasti parempia tuloksia. Vertex AI -ratkaisuissa on edelleen huomattavaa potentiaalia, mutta niiden tehokkuus riippuu olennaisesti syötettävän datan laadusta ja rakenteesta.

Pilvipalveluiden hyödyntämisen osalta opittiin paljon. GCP ja Grafana Cloud tarjosivat tehokkaan, skaalautuvan ja nopeasti käyttöönotettavan alustan – mutta samalla kustannusten seuranta nousi esiin tärkeäksi osa-alueeksi. Vaikka opinnäytetyössä hyödynnettiin Google Cloudin koekäyttöetua (noin 300 €) ja Grafana Cloudin edullista 19 euron kuukausilisenssiä, todellisessa tuotantoympäristössä kustannukset voivat nousta nopeasti, jos dataa säilytetään pitkään tai jos mallien ajotiheyttä ei optimoida. On suositeltavaa kehittää organisaation sisäiset käytännöt esimerkiksi datan arkistointiin, tarpeettomien kyselyiden karsimiseen ja mallien käyttöönoton suunnitteluun. Pilvipalveluilla on kiistattomat hyödyt, mutta tuotantokäytössä myös merkittävät kustannukset, eikä tätä voi unohtaa yrityksenkään näkökulmasta.

Kehitystyön aikana todettiin, että suunnittelun puutteet ja aikataulujen venyminen olivat selkeitä haasteita. Työ rönsyili välillä useisiin eri vaihtoehtoihin ja kokonaisuuden hallinta oli hetkittäin haastavaa. Tämä korostaa tarvetta suunnitelmalliselle lähestymistavalle ja tiukemmalle projektihallinnalle etenkin silloin, kun kehitetään teknisesti vaativaa ratkaisua. On kuitenkin myönnettävä, että kokeilujen kautta opittiin paljon – ja monet lopulliset ratkaisut perustuivat

iteratiiviseen kehittämiseen ja käytännön testaamiseen, mikä on arvokas osa tällaisessa kehitysprojektissa.

On myös hyvä huomioida, että pankkien välisten rahansiirtojen analysointi on luonteeltaan erityisen vaikea kohde ennustamiselle. Pankit ovat itsenäisiä organisaatioita, joiden huoltokatkoista tai teknisistä ongelmista ei ole julkista reaaliaikaista tietoa. Tämä vaikeuttaa mallien tarkkaa kouluttamista ja tekee ennustamisesta haastavaa, ellei jopa mahdotonta tietyissä tilanteissa. Tästä huolimatta rahansiirtojen valvonta nähtiin tärkeänä toimintona ja siihen haluttiin nimenomaan soveltaa koneoppimista, koska samaa valvontakonseptia voidaan tulevaisuudessa laajentaa myös muihin kriittisiin prosesseihin. On myös rohkaisevaa, että vaikka tuotantoympäristö käyttää omia työkalujaan, tämä opinnäytetyö osoitti, että vastaava ratkaisu voidaan toteuttaa kevyemminkin, omassa kehitysympäristössä. Tämä mahdollisti vapauden testata erilaisia lähestymistapoja ilman byrokraattisia rajoitteita.

Uusien versioiden, kuten Grafana 12:n, mukana tulevat ominaisuudet, esimerkiksi parannettu käyttöoikeushallinta, kehittyneemmät visualisointielementit ja laajemmat integraatiomahdollisuudet – avaavat tulevaisuudessa uusia mahdollisuuksia entistä tehokkaampaan valvontaratkaisuun. Nämä mahdollistavat entistä paremman käytettävyyden ja laajennettavuuden, mikä tukee mallien laajempaa hyödyntämistä myös yrityksen muissa liiketoimintaprosesseissa.

Yhteenvetona voidaan todeta, että tämän työn puitteissa kehitetty järjestelmä tarjoaa vahvan perustan tuotantotason ML-valvonnalle. Jatkokehitystä suositellaan erityisesti Vertex AI:n mallien osalta, ja lisäksi pilvikustannusten hallintaan kannattaa kiinnittää erityistä huomiota, kun siirrytään laajempaan ja jatkuvampaan tuotantokäyttöön.

LÄHTEET

Aggarwal, C. C. (2017). Outlier analysis (2nd ed.). Springer.

Alpaydın, E. (2020). Introduction to machine learning (4th ed.). The MIT Press.

Anttila, J. (2024). Raideliikenteen riskiperusteisen valvonnan kehitystyö [Bachelor's thesis, Theseus]. http://www.theseus.fi/handle/10024/866958

Dehnavi, N. (2025). Machine learning algorithms and visualization techniques for financial anomaly detection [Bachelor's thesis, Theseus]. https://www.theseus.fi/handle/10024/893775

Fu, H., & Qi, K. (2022). Evaluation Model of Teachers' Teaching Ability Based on Improved Random Forest with Grey Relation Projection. Scientific Programming, 2022, 1–12. https://doi.org/10.1155/2022/5793459

Google. (2025a). BigQuery documentation. https://cloud.google.com/bigquery/docs

Google. (2025b). Dataflow overview. https://cloud.google.com/dataflow/docs/overview

Google. (2025c). Forecasting with AutoML | Vertex AI. https://cloud.google.com/vertex-ai/docs/tabular-data/forecasting/overview

Google. (2025d). Python client library | Google Cloud. https://cloud.google.com/python/docs/reference/aiplatform/latest

Grafana Labs. (2025a). Google BigQuery plugin for Grafana. https://grafana.com/grafana/plugins/grafana-bigquery-datasource/

Grafana Labs. (2025b). Grafana Alerting | Grafana documentation. https://grafana.com/docs/grafana/latest/alerting/

Grafana Labs. (2025c). Time series | Grafana documentation. https://grafana.com/docs/grafana/latest/panels-visualizations/visualizations/time-series/

Grafana Labs. (2025d). What's new in Grafana | Grafana documentation. https://grafana.com/docs/grafana/latest/whatsnew/

Géron, A. (2023). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems (3rd ed.). O'Reilly.

Hyndman, R. J., & Athanasopoulos, G. (2021). Forecasting: Principles and practice (3rd ed.). OTexts. https://otexts.com/fpp3/

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: With applications in R. Springer.

Kulmala, T. (2024). Tekoäly osana kodin kameravalvontaa [Bachelor's thesis, Theseus]. http://www.theseus.fi/handle/10024/854158

Lässig, F. (2021, July 9). A guide to building a financial transaction anomaly detector. Unit8. https://unit8.com/resources/a-guide-to-building-a-financial-transaction-anomaly-detector/

McKinney, W. (2022). Python for data analysis: Data wrangling with pandas, NumPy, and Jupyter (3rd ed.). O'Reilly.

OpenAI. (2025). Random Forest -algoritmin toimintakaavio. [Generoitu kuva]. OpenAI ChatGPT.

Raghavendra, S. (2023). Beginner's Guide to Streamlit with Python: Build Web-Based Data and Machine Learning Applications (1st ed.). Apress.

scikit-learn developers. (2025). IsolationForest. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html

Soria, J. J., Loayza Abal, R., & Segura Peña, L. (2024). Machine learning models for money laundering detection in financial institutions: A systematic literature review. Proceedings of the 22nd LACCEI International Multi-Conference for Engineering, Education and Technology. https://doi.org/10.18687/LACCEI2024.1.1.1682