Mikko Wiseman - mikko.wiseman@edu.turkuamk.fi

ICT-alan täydennyskoulutus, Turun Ammattikorkeakoulu, kevät 2020

Sovellusohjelmointi

Projektisuunnitelma

16.2.2020

**Data-analyysityökalu yrityksen saatavien analysointiin taloushallinnon Accounts Receivable -ympäristössä Python-ohjelmointikieltä käyttäen**

**TAUSTAA:**

Accounts Receivable (AR, myyntireskontra) on vastuussa yrityksen saatavista sen asiakkailta. Nämä saatavat ovat yleensä yrityksen pääasiallinen tulonlähde, joten kassavirran jatkuva analysointi on tärkeää yrityksen talouden tasapainon varmistamiseksi. Yrityksen saatavia seuraamalla voidaan rakentaa ennuste kassavirran kehittymisestä. Lisäksi saatavien analysointi on tarpeellista nk. huonojen velallisten identifioimiseksi ja mahdollisiin maksumuistutus- ja perintätoimenpiteisiin ryhtymistä varten.

Taloushallinnon käytössä on useita vakiintuneita ohjelmistopaketteja, mm. SAP, Sage ja suomalainen Visman erityisesti PK-sektorille suunnattu ePasseli. Monien näiden ohjelmistojen graafiset visualisointiominaisuudet ovat kuitenkin rajalliset ja apuna on käytettävä muita integroimattomia ohjelmistoja, kuten esimerkiksi Microsoft Excelin diagrammit, jotka kuitenkin vaativat tietojen siirtämistä Exceliin ja diagrammien rakentamista Excelin graph-työkalulla. Menetelmän heikkoutena on manuaalisen työn määrä (esimerkiksi allamainitun aging schedule -kategorioiden manuaalista määrittelyä). Erilaisten graafisten visualisointien käyttö nopeuttaa ja havainnollistaa analysointia ja mahdollistaa mm. ennusteiden laatimisen. Mm. ePasseli-ohejelmistosta puuttuu ominaisuus aging schedule ja bad debt -diagrammien tuottamiseen.

Olen valinnut projektille Accounts Receivable -viitekehyksen erityisesti sen vuoksi, että olen työskellellyt useita vuosia taloushallinnon tehtävissä, joten toimiympäristö ja automatisoidun data-analyysin tarpeet sen kontekstissa ovat minulle tuttuja.

**TOTEUTETTAVAN SOVELLUKSEN KUVAUS JA TOTEUTUSTAPA:**

Projektissa toteutettava sovellus vastaa edellämainittuihin tarpeisin tarjoamalla AR-ympäristön havainnollisen data-analyysi- ja visualisointityökalun saatavien seurantaan. Sovellus lukee yrityksen laskustusjärjestelmässä olevien myyntilaskujen tiedot CSV-muotoisesta tiedostosta sovelluksen analysoitavaksi. CSV-tiedostojen luku mukautetaan vastaamaan erityisesti Visman ePasseli-myyntilaskuraporttien formaattia.

Sovellus rakennetaan käyttämällä Pythonin numPy, Matplotlib ja Pandas -kirjastoja. Graafinen käyttöliittymä rakennetaan pääosin TkInter-kirjastoa käyttämällä. Käyttäjä ohjaa ohjelmaa sovelluksen painikkeilla ja syöttämällä tarvittaessa tietoja sovelluksen antamiin kenttiin. Sovelluksen tuottamat raportit esitetään käyttöliittymän ikkunassa diagrammeina ja listauksina. Tarpeen ja mahdollisuuksien mukaan sovellukseen lisätään ominaisuudet raporttien tallentamiseen CSV-tiedostoissa.

**KÄYTETTÄVÄT DATA-ANALYYSIMENETELMÄT:**

Sovelluksen ladattua tiedot myyntilaskuista sovellus esittää yhteenvedon jo saaduista maksusuorituksista, tulevista ja erääntyneistä saatavista sekä keskimääräisestä maksuajasta (days sales outstanding). Sovelluksen tarjoamat analyysit voidaan toteuttaa joko kaikista järjestelmässä olevista laskuista tai vain tietyn yrityksen kohdalta.

Sovelluksessa käytetään lisäksi seuraavia kolmea AR-ympäristössä yleisesti käytettyjä data-analyysimenetelmää:

1. Trend line -analyysi saatavien ennustamiseksi

Yrityksen saatavien ennustetta ja toteutumista voidaan seurata trend line -analyysillä tulevien saatavien erääntymisestä (ennuste) ja toteutuneesta maksamisesta (aktuaalinen tilanne). Sovelluksen kassavirran lineaarisessa mallinnuksessa oletetaan, että yrityksen laskut erääntyvät suhteellisen tasaisesti kuukauden eri päivinä eikä esimerkiksi kuukauden ensimmäisenä päivänä, kuten on kyse esimerkiksi tiettyjen jatkuvien palveluiden laskutuksen tapauksessa.

1. Aging schedule -diagrammi eräpäivän ylittäneiden saatavien määrän tarkastelemiseen

Aging schedule -diagrammissa jaotellaan eräpäivän ylittäneet saatavat kategorioihin niiden erääntymisen asteen mukaan. Taloushallinnossa yleisesti käytettävät kategoriat ovat 0-30 päivää, 31-60 päivää ja 61+ päivää.

1. Bad debt -raportti erääntyneiden saatavien analysoinniksi

Bad debt -raportilla on tärkeä merkitys yritykselle luottokelpoudeltaan heikkojen asiakkaiden identifioimiseksi, riskiarvion laatimiseksi ja mahdollisia jatkotoimenpiteitä varten. Raportti esittää tiedot erääntyneistä laskuista velallisen toiminimen, velan määrän ja eräpäivän ylittämiskauden pituuden mukaan. Sen tietoja voidaan käyttää hyväksi päätettäessä jatkotoimenpiteistä (mm. luottokelpoisuuden arviointi ja perintätoimenpiteistä päättäminen). Bad debt-raportti vertailee myös erääntyneiden laskujen suhdetta kokonaissaataviin eri velallisten osalta, joka auttaa velallisten luottokelpoisuuden määrittelemisessä.

**OPPIMISTAVOITTEET:**

1. Python-ohjelmointikielen hallinta ja käyttö data-analyysissä ja datan visualisoinnissa.
2. Taloushallinnon data-analyysitarpeiden (Accounts Receivable) ymmärtäminen, mallintaminen ja toteuttaminen käytännössä Python-sovelluksessa.
3. GUI-pohjaisen sovelluksen rakentaminen käyttäjäystävällisestä ja havainnollisesta lähtökohdasta.

**OPPIMISMATERIAALIT:**

Idris, I. 2014. Learning NumPy Array. Birmingham: Packt Publishing.

Idris, I. 2014. Python Data Analysis. Birmingham: Packt Publishing.

Toivonen, J. 2019. Data Analysis with Python. Helsingin Yliopisto. Available at: <https://saskeli.github.io/data-analysis-with-python-summer-2019/index.html>

**AIKATAULU:**



Projektin loppuyhteenveto ja itsearviointi

Wiseman Mikko 28.04.2020 15:25

Sovelluksen kehitys ja henkiökohtainen oppimisprosessi:  
Lähdin kehittämään sovellusta taloushallinnon luotonvalvonnan analyysin työkaluksi sen kokemuksen pohjalta, mitä minulla on taloushallinnon työtehtävissäni em. tehtävistä kertynyt. Tarkoitukseni oli siis lähteä kehittämään sovellusta tosielämän - "asiakkaan" - tarpeiden pohjalta mahdollisimman käytännönläheisesti. Minulla ei ollut aikaisempaa koodausosaamista Pythonin puolelta. Lähdinkin siis aluksi opettelemaan Pythonin alkeita käyttäen hyväksi mm. SoloLearn -sovelluksen oppitunteja ja w3schools-harjoituksia. Aiempi koodauskokemukseni Basicin, HTML:n ja Javan puolelta osoittautui erittäin hyväksi avuksi tässä prosessissa ja Pythonin looginen, luonnollisen kielen omainen syntaksi teki koodaamisesta helppoa. Varsinainen tietojen lukeminen, luokittelu, verifiointi ja analysointi oli suhteellisen helppoa Javan puolelta; lähinnä lisätöitä teetti Pythonin GUI-luokkien ja metodien käyttö. Havaitsin itselleni parhaan tavan oppia tutkimalla jo joku olemassaoleva mallisovellus tai koodinpätkä aiheesta, jonka osaset ja niiden toiminnan analysoin. Näin pystyin siirtämään oppimani toiminnallisuudet omaan sovellukseeni.  
  
Käytännössä oppimista tapahtui koko projektin ajan. Jouduin konsultoimaan Pythonin metodidokumentaatiota jatkuvasti, koska esimerkiksi Tkinterin ja Matplotlibin kirjastot ovat paikoitellen monimutkaisia. Törmäsin myös muutamaan Pythonin versio 3.6:n bugiin esimerkiksi Treeview-metodin rivinvärien muuttamisessa, mitkä vaativat selvitystyötä ja erityisratkaisuja niiden korjaamiseksi.  
  
Omalta osaltani merkittävintä oppimista tapahtui Python-koodauksen opettelemisen lisäksi GUI-toteutuksen alueella. Erilaisten listojen, graafien ja painikkeiden sijoittelu järjestelmäikkunoihin oli alue, jolla minulla oli ollut varsin vähän kokemusta vähäisen Java Swing ja FX-kokeilun lisäksi. Siitä lienee hyötyä tulevissa front end -projekteissa siitäkin huolimatta, että ePasselin tapaiset SaaS-sovellukset käyttävätkin front end -puolella. Peruskäsitteiden oppimiseen Pythonin Tkinter -kirjasto tarjosi kuitenkin hyvät puitteet huolimatta sen rajoituksista visuaalisuuden suhteen. Kaupallisen sovelluksen toteuttamiseen olisinkin käyttänyt todennäköisesti muita, pitkälle kehitetympiä ohjelmakirjastoja ja lisäosia.  
  
Koska Sovellusohjelmoinnin kurssi kulki omassa opintosuunnitelmassani Ohjelmistosuunnittelun- ja mallintamisen kurssin rinnalla, pystyin käyttämään kummallakin kurssilla opittua hyödykseni sovelluksen suunnittelussa. Niinpä esimerkiksi ohjelmistokaavion ja käyttäjätarinoiden laatimisesta, versionhallinnasta, testaamisesta ja Git-repojen käytöstä oli hyötyä sovellusta tehtäessä. Tietokantojen kurssilta hyödynsin relaatiotietokantojen mallinnusta ja toteutusta, vaikka valmis sovellus ei olekaan yhteydessä varsinaisesti mihinkään tietokantaan.  
  
Alkuperäinen tarkoitukseni oli käyttää numpy-arrayta enemmän tietojen analysointiin. Kuitenkin analysoinnissa käytettävän tiedon monimuotoisuus (integer, float, datetime, ja string-tyyppistä dataa sekaisin) johti perinteisen olio-ohjelmoinnin menetelmien käyttöön tiedon tallennuksessa. Havaitsin myös laskudatan tallentamisen erillisinä laskuolioina paremmaksi tietojen eheyden kannalta, sillä se mahdollisti suorat viittaukset olion muuttujiin ja siten vähensi riskiä tietoalkioiden sekoittumiseen keskenään.  
  
Valmis sovellus vastasi melko tarkasti alkuperäisessä projektisuunnitelmassani hahmottelemaa sovellustani. Olisin, mikäli mahdollista, halunnut implementoida vielä lisää analyysitoiminnallisuuksia sovellukseeni ja hyödyntää koneoppimista ja tekoälyä esimerkiksi luottokelpoisuuden analysoinnissa. Kuitenkin käytössä olevan laskudatan rajallisuus asetti rajoituksia tälle: todelliselle koneoppimiselle tarvittaisiin paljon isompia "todellisen maailman" datajoukkoja. Koska tällainen data on kuitenkin ns. business sensitive -tietoa enkä tuntunut löytävän keinotekoisia valmiita datajoukkoja netistäkään jouduin tekemään joukot itse Visman ePasseli -järjestelmän sample datan pohjalta ja testaamaan sovellustani niillä.  
  
Valmis sovellukseni on hyvin laajennettava: data voitaisiin helposti lukea esimerkiksi SQL tai NoSQL -tietokannoista, mutta koska minulla ei ollut pääsyä mihinkään olemassaolevaan järjestelmään eikä siten tietoa näiden tietokantojen arkkitehtuurista en niitä tähän sovellukseen voinut sellaisenaan lisätä. Halutessani olisin voinut lisätä tiedoille tallennusominaisuuden esimerkiksi MongoDB-tietokantaan. Toisaalta tällaisessa tiedon tallennuksessa olisi pitänyt käyttää enemmän resursseja tietoturvallisuudesta huolehtimiseen.  
  
Sovellukseeni jäi toki paljon parannettavaa esimerkiksi luokkien käytön suhteen: prosesseja olisi voinut eriyttää tehokkaammin eri luokkiin ja metodeihin koodin laadun ja laajennettavuuden parantamiseksi. Olen kuitenkin varsin tyytyväinen lopputulokseen ottaen huomioon, että lähdin Python-koodauksesta ns. nollasta ilman juuri minkäänlaista tietoa kielen käytöstä. Onnistuin kurssin aikana kehittämään Python-taitojani merkittävästi ja samalla laajentamaan ymmärrystäni sovellusten toteuttamisestaa olio-ohjelmointia hyväksikäyttäen.  
  
Koin sovelluksen rakentamisen kaikenkaikkiaan erittäin mielekkäänä ja mukaansatempaavana työnä. Koodin kirjoittaminen tuntui tulevan minulta helposti, tuntui mielenkiintoiselta ja mielekkäältä, ja olen siksi entistä varmistuneempi siitä, että sovelluskehitys on omalta kohdaltani mielekäs ja järkevä uraratkaisu.  
  
Keskeiset oppimisalueet:  
- Python 3 -ohjelmointi sen alkeista lähtien ja kielen eroavaisuuksien sisäistäminen mm. C-tyyppisen paradigman kieliin (esim. Java) verrattuna  
- Olio-ohjelmointi, luokkien käyttö tietojen tallentamiseen ja jäsentämiseen, metodien käyttö toimintojen tehokkaaseen järjestämiseen  
- Sovelluksen suunnittelu käyttötarpeiden pohjalta  
- GUI-suunnittelu ja toteutus erityisesti Tkinteriä käyttäen, UX-näkökulma suunnitteluun  
- Tietojen luku ja verifiointi tiedostosta  
- Erilaiset tilastolliset analyysimenetelmät ja niiden implementointi ohjelmoinnissa  
- Ohjelmiston suunnittelu laajempana kokonaisuutena  
- Versionhallinta  
- Virhetilanteiden huomioonottaminen koodia suunniteltaessa, käyttäjävirheiden eliminointi suunnitteluvaiheessa, exception and error handling  
- Ohjelmiston testaus  
  
Ajankäyttö:  
Projektiin käytettävää aikaa on vaikea arvioida tarkasti, koska alussa kului paljon aikaa Pythonin opetteluun ja äitini äkillinen sairastuminen ja kuolema keskeytti työnteon noin kuukaudeksi maaliskuun alussa. Huhtikuusta alkaen koodasin sovellusta lähes päivittäin 6-10 tuntia päivässä, joten arvioin kokonaistyömäärän noin 150 tunnin kieppeille, johon sisältyy Python-ohjelmoinnin opettelu. Sovellus valmistui 27.4.2020.  
  
Oppimisresurssit:  
Idris, I. 2014. Learning NumPy Array. Birmingham: Packt Publishing.  
Idris, I. 2014. Python Data Analysis. Birmingham: Packt Publishing.  
Toivonen, J. 2019. Data Analysis with Python. Helsingin Yliopisto. Available at: <https://saskeli.github.io/data-analysis-with-python-summer-2019/index.html>  
www.w3schools.com -sivuston Python-harjoitukset  
SoloLearn -iPhone applikaation Python-kurssi  
docs.python.org - hyödyllinen erityisesti metodien speksejä rakennettaessa  
<https://www.geeksforgeeks.org/python-gui-tkinter/> - yhtenä monista GUI-rakentamisen resursseista  
<https://www.tutorialspoint.com/python/> - täältä löytyi paljon hyödyllisiä esimerkkejä  
www.github.com - olemassaolevia Python-repoja analysoimalla opin paljon Python-ohjelmoinnin käytänteistä