Google Advanced Data Analytics Capstone projektas

Šis dokumentas yra skirtas pristatyti projekto procesą, mano mintis, kaip buvo sprendžiamos problemos. Dokumento struktūra seka chronologinę projekto eigą. „Capstone“ projektas yra skirtas kurso mokiniams pritaikyti įgytas žinias realaus pasaulio situacijoje. Atlikdamas kurso finalinę užduotį padariau klaidų, praleidau kai kuriuos žingsnius, bet jas ištaisiau ir šiame apraše pateikiau pataisytą užduotį su komentarais apie padarytas klaidas. Čia galite rasti visą projektui naudotą python kodą, vizualizacijas. Kodas yra nukopijuotas teksto formatu tiesiai iš projektui atlikti naudotos Jupyter Notebook platformos. Kodo atsakai, vizualizacijos pateikiami nuotraukose.

Duomenys ir jų tyrimas (angl. Exploratory Data Analysis, EDA)

Duomenų stulpelių paaiškinimas	
satisfaction_level	Darbuotojų pasitenkinimo darbu lygis [0-1]
last_evaluation	Paskutinio darbuotojo veiklos vertinimo balas [0-1]
number_project	Projektų, prie kurių įgyvendinimo darbuotojas prisideda, skaičius
average_monthly_hours	Vidutinis darbuotojo dirbtų valandų skaičius per mėnesį
time_spend_company	Kiek laiko darbuotojas dirba įmonėje (metais)
Work_accident	Ar darbuotojas patyrė nelaimingą atsitikimą darbe, ar ne
left	Ar darbuotojas paliko įmonę, ar ne
promotion_last_5years	Ar darbuotojas buvo paaukštintas per pastaruosius 5 metus
Department	Darbuotojo departamentas
salary	Darbuotojo atlyginimas (USD)

Įkėliau reikalingus paketus.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV, train_test_split
from sklearn.metrics import accuracy_score, precision_score, recall_sco
re,\
f1_score, confusion_matrix, ConfusionMatrixDisplay, classification_repo
rt
```

Priskyriau duomenis kintamajam df0 ir .head funkcijos pagalba apžiūrėjau kelias eilutes.

```
df0 = pd.read_csv("HR_capstone_dataset.csv")
df0.head()
```

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	Department	sa
0	0.38	0.53	2	157	3	0	1	0	sales	
1	0.80	0.86	5	262	6	0	1	0	sales	med
2	0.11	0.88	7	272	4	0	1	0	sales	med
3	0.72	0.87	5	223	5	0	1	0	sales	
4	0.37	0.52	2	159	3	0	1	0	sales	

```
df0.info()
```

Ši funkcija pateikė duomenų tipus.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

```
df0.describe()
```

Funkcija pateikė statistinius duomenis: stebėjimų skaičių, vidurkį, kvartilius ir kt.

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	0.021268
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	0.144281
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	0.000000
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	0.000000
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	0.000000
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	0.000000
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	1.000000

```
df0 = df0.rename(columns={'Work_accident': 'work_accident',
                          'average_montly_hours': 'average_monthly_hours',
                          'time_spend_company': 'tenure',
                          'Department': 'department'})
df0.columns
```

Užduotis rekomendavo ištaisyti ir sutvarkyti stulpelių pavadinimus į snake_case formatą.

```
Index(['satisfaction_level', 'last_evaluation', 'number_project',
       'average_monthly_hours', 'tenure', 'work_accident', 'left',
       'promotion_last_5years', 'department', 'salary'],
      dtype='object')
```

Patikrinau ar duomenys turėjo trūkstamų verčių. Jų nebuvo. Jei trūkstamų verčių būtų pasitaikę, būčiau jas pašalines .drop funkcija

```
df0.isna().sum()
```

```
satisfaction_level       0
last_evaluation          0
number_project           0
average_monthly_hours    0
tenure                   0
work_accident            0
left                     0
promotion_last_5years    0
department               0
salary                   0
dtype: int64
```

```
df0.duplicated().sum()
```

Patikrinau ar duomenyse buvo pasikartojančių duomenų. Rezultatas parode 3008 pasikartojančias eilutes.

```
df0[df0.duplicated()].head()
```

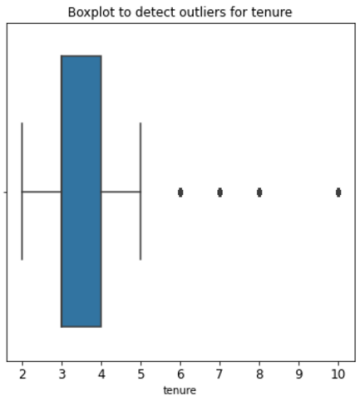
Vizualiai pažiūrėjęs pasikartojančias eilutes nusprendžiau, kad jos yra netinkamos, kad jų pasikartojimas yra klaidingas.

```
df1 = df0.drop_duplicates(keep='first')
```

Todėl atsikračiau pasikartojančių eilučių, o gautą rezultatą priskyriau kitam kintamajam df1.

Sukūriau boxplot vizualizaciją, patikrinti ar „Tenure“ stulpelyje egzistuoja išsiskiriantys stebėjimai (outliers). Matome, kad yra stebėjimų, kurie išeina už dešiniojo ūso(whisker), todėl vizualizacija rodo, kad jų esama, tačiau pasirinkau šių stebėjimų nepašalinti. Nusprendžiau, kad šiuo atveju išsiskiriantys stebėjimai turi reikšmę ir yra reikalingi reprezentatyvaus modelio kūrimui. Tolimesniame žingsnyje šio sprendimo teko atsisakyti.

```
plt.figure(figsize=(6,6))
plt.title('Boxplot to detect outliers for tenure', fontsize=12)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
sns.boxplot(x=df1['tenure'])
plt.show()
```



Vizualizacijos

Pasitelkiant vizualizacijas bandžiau atrasti įžvalgas ar ryšius tarp kintamųjų. Kai kurios vizualizacijos čia neįtrauktos ir jų kodo neišsaugojau, nes jos paprasčiausiai nebuvo labai naudingos ir, mano manymu, nesuteikė įdomių įžvalgų.

```
plt.figure(figsize=(16, 9))
sns.scatterplot(data=df1, x='average_monthly_hours', y='satisfaction_le
vel', hue='left', alpha=0.4)
plt.axvline(x=166.67, color='#ff6361', label='166.67 hrs./mo.', ls='--'
)
plt.legend(labels=['166.67 hrs./mo.', 'left', 'stayed'])
plt.title('Monthly hours by satisfaction level', fontsize='14');
```

Sukūriau grafiką, kuris pavaizduotų ryšį tarp darbuotojų pasitenkinimo ir per darbo valandų per mėnesį. Vizualizacijos pavadinime įsivėlė klaida – pavadinimas turėtų būti 'Monthly hours by satisfaction level'. Pasinaudojau užduoties rekomendacija įtraukti raudoną punktyrinę liniją, vaizduojančia vidutį darbo valandų per mėnesį lygi (166.67h.), kai dirbama 40 val. per savaitę. Galima pastebėti ryši tarp žemo pasitenkinimo lygio ir didelio darbo valandų kiekio. Tai ypač matoma didelėje grupėje, kuri paliko darbo vietą.

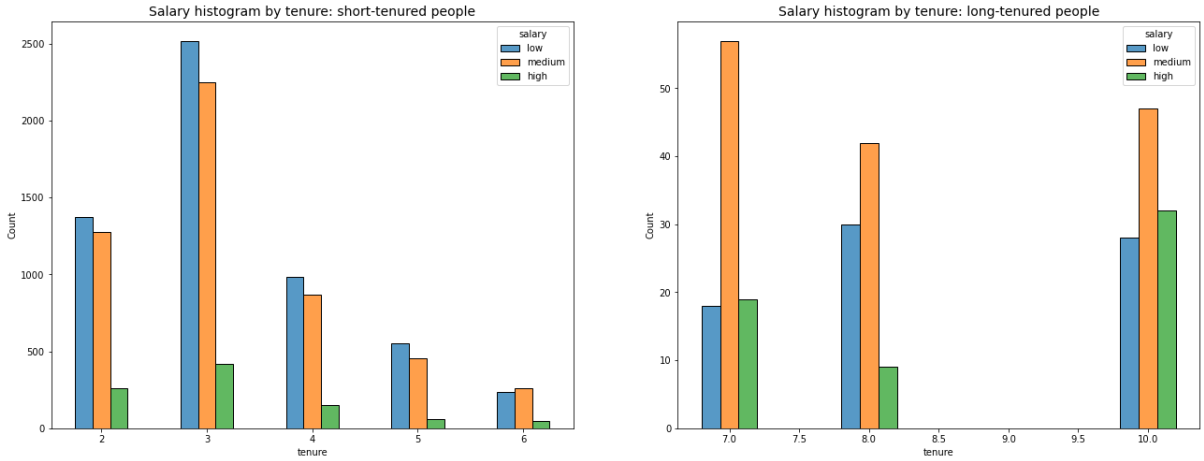
Duomenų pasiskirstymas man pasirodė nelogiškas. Didelis kiekis darbuotojų, dirbančių gerokai virš 40 val. per savaitę, įvertina savo pasitenkinimo lygį virš 0,5. Apsvarstydamas šį faktą praleidau daugiau laiko nei norėjau. Paieškojęs informacijos radau paaiškinimą, kad šis pastebėjimas yra teisingas ir pasak užduoties autorių, indikuoja suklastotus duomenis. Man nepatiko užduoties sudarytojų pasirinkimas parinkti tokius duomenis. Juk sufabrikuoti stebėjimai negali sudaryti teisingas įžvalgas pateikiantį regresijos modelį.


```
fig, ax = plt.subplots(1, 2, figsize = (22,8))
tenure_short = df1[df1['tenure'] < 7]
tenure_long = df1[df1['tenure'] > 6]
sns.histplot(data=tenure_short, x='tenure', hue='salary', discrete=1,
```

```
            hue_order=['low', 'medium', 'high'], multiple='dodge', shr
ink=.5, ax=ax[0])
ax[0].set_title('Salary histogram by tenure: short-tenured people', fon
tsize='14')
sns.histplot(data=tenure_long, x='tenure', hue='salary', discrete=1,
            hue_order=['low', 'medium', 'high'], multiple='dodge', shr
ink=.4, ax=ax[1])
ax[1].set_title('Salary histogram by tenure: long-tenured people', font
size='14');
```

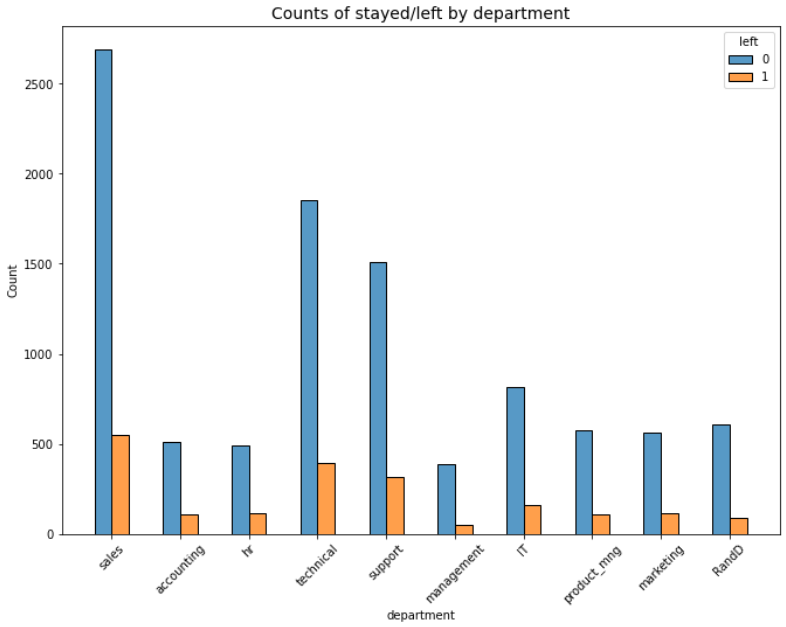
Nusprendžiau sudaryti stulpelines diagramas, kurios pavaizduotų darbo užmokesčio klasifikacijų (low, medium, high) pasiskirstymą pagal metus praleistus dirbant įmonėje. Bandydamas sudaryti šį grafiką neišskyriau „tenure“ į „long“ ir „short“ dalis, tačiau tai pakeičiau pamatęs, kad užduoties kūrėjų pasiūlymas paverčia grafikus tinkamesniais tolimesniam naudojimui.

Galima pastebėti, kad daug įmonėje išdirbę darbuotojai nėra neproporcingai daugiau apmokami. Trūksta atvejų darbuotojų, kurie būtų išdirbę tarp 8,5 ir 10 metų.



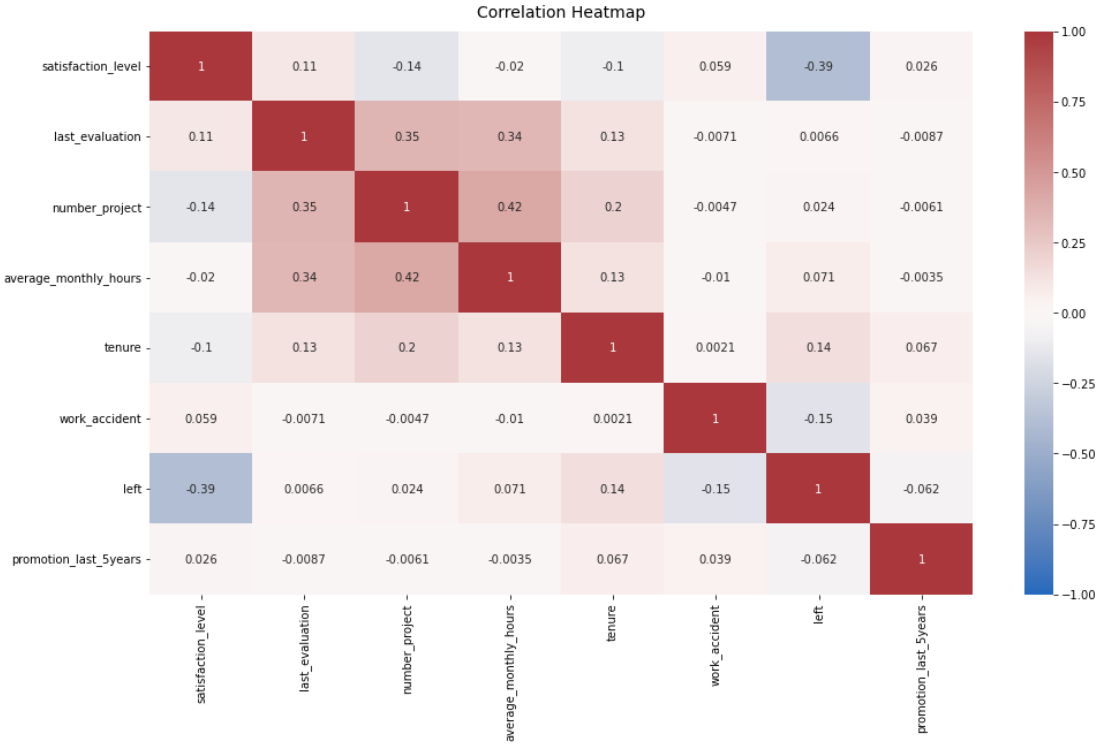

```
plt.figure(figsize=(11,8))
sns.histplot(data=df1, x='department', hue='left', discrete=1,
            hue_order=[0, 1], multiple='dodge', shrink=.5)
plt.xticks(rotation='45')
plt.title('Counts of stayed/left by department', fontsize=14);
```

Toliau pavaizdavau darbuotojų išėjimo iš ir pasilikimo įmonėje histogramą, išskirstytą pagal departamentą. Pardavimų, technikos ir konsultavimo skyriai turi daugiausia darbuotojų. Nepastebiu reikšmingo skirtumo tarp departamentų, todėl negaliu teigti, jog vienas ar kitas skyrius turi darbuotojų išėjimo problemą.




```
heatmap = sns.heatmap(df0.corr(), vmin=-1, vmax=1, annot=True, cmap=sns
.color_palette("vlag", as_cmap=True))
heatmap.set_title('Correlation Heatmap', fontdict={'fontsize':14}, pad=
12);
```

Galiausiai sudariau „Heat Map“ grafiką, vaizduojantį koreliaciją tarp kintamųjų. Matoma teigiama koreliacija tarp projektų skaičiaus ir darbo valandų per mėnesį, t.y. didėjant šiems rodikliams didėja tikimybė, kad darbuotojas išeis. Ta pati koreliacija matoma ir su darbuotojo įvertinimo bei darbo įmonėje metų. Šios koreliacijos ekonominė logika galėtų būti ta, kad darbuotojai praleidę daug metų įmonėje ir gavę aukštus įvertinimus, palieka įmonę dėl karjeros galimybių. Įmonei derėtų pagalvoti ar būtu verta imtis priemonių skatinti šiuos darbuotojus pasilikti. Matoma aiški neigiama koreliacija tarp pasitenkinimo lygio ir išėjimo. Šio rodiklio įvertis nestebina.



Modelio sudarymas

Pirmiausia reikėjo pažvelgti į kintamąjį, kurį reikėjo prognozuoti – darbuotojų išėjimas. Šis kintamasis gali turėti reikšmes 0 arba 1, t.y. jis dvireikšmis (angl. binary). Dvireikšmiams kintamiesiems prognozuoti GADA kursas mokė naudoti logistinę regresiją. Todėl pasirinkau būtent ją.

Kad galėčiau sudaryti modeli reikėjo kokybinius kintamuosius pasiversti kiekybiniais. Fiktyvius kintamuosius departamentams sukurti sunku nebuvo. Tačiau atlyginimo kintamasis savyje talpina hierarchija, todėl čia man kilo problemų. Galiausiai atradau atsakymą ir paverčiau atlyginimo kintamąjį į vertes [0; 2].

```
df_enc = df1.copy()

df_enc['salary'] = (
    df_enc['salary'].astype('category')
    .cat.set_categories(['low', 'medium', 'high'])
    .cat.codes
)

df_enc = pd.get_dummies(df_enc, drop_first=False)

df_enc.head()
```

Sekančiame žingsnyje panaikinau anksčiau minėtus išsiskiriančius duomenis. Kurso autoriai pabrėžė, jog logistinė regresija yra ganėtinai jautri išsiskiriantiems stebėjimams. Todėl teko pripažinti klaidą ir šiuos stebėjimus pašalinti.

Apskaičiuotą tarpkvartilinį intervalą panaudojau viršutiniam ir apatiniam limitui apibrėžti. Visos vertės, kurios yra aukščiau ar žemiau minėtų intervalų, nebuvo priskirtos į naujus duomenis df_logreg

```
percentile25 = df1['tenure'].quantile(0.25)
percentile75 = df1['tenure'].quantile(0.75)

iqr = percentile75 - percentile25

upper_limit = percentile75 + 1.5 * iqr
lower_limit = percentile25 - 1.5 * iqr
print("Lower limit:", lower_limit)
print("Upper limit:", upper_limit)
df_logreg = df_enc[(df_enc['tenure'] >= lower_limit) & (df_enc['tenure'
] <= upper_limit)]
```

Tuomet y kintamajam priskyriau priklausomą kintamąjį.

```
y = df_logreg['left']
```

X kintamajam priskyriau visus likusius kintamuosius.

```
X = df_logreg.drop('left', axis=1)
```

Tuomet padalinau pasirinktus duomenis į mokymo ir testavimo dalis. Šiam modeliui mokyti bus naudojami 75% atsitiktinai parinktų duomenų. Likę 25% bus naudojami regresijos testavimui.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2
5, stratify=y, random_state=0)
```

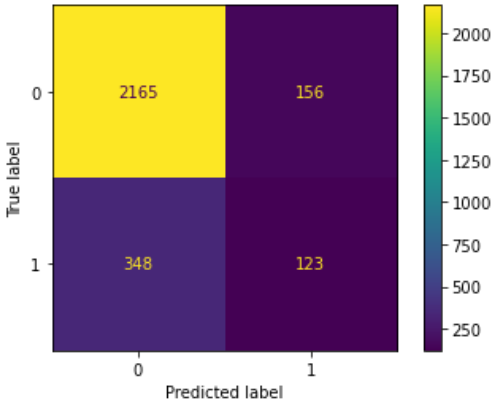
Tada sudariau patį modelį. Tolimesniam darbui modelio prognozes priskyriau kintamajam y_pred.

```
log_clf = LogisticRegression(random_state=42, max_iter=500).fit(X_train
, y_train)
y_pred = log_clf.predict(X_test)
```

Modelio rezultatams vizualiai įvertinti naudojau painiavos matricą(confusion matrix). Viršutiniame kairiajame langelyje rodomas tikrų neigiamų(true negatives) rezultatų skaičius. Viršutiniame dešiniajame langelyje - klaidingai teigiamų(false positives) rezultatų skaičius. Apatiniame kairiajame langelyje - klaidingų neigiamų(false negatives) rezultatų skaičius. Apatiniame dešiniajame langelyje – teigiamų(true positives) rezultatų skaičius.

```
log_cm = confusion_matrix(y_test, y_pred, labels=log_clf.classes_)
log_disp = ConfusionMatrixDisplay(confusion_matrix=log_cm,
                                  display_labels=log_clf.classes_)
log_disp.plot(values_format='')
```

```
plt.show()
```



Autorių rekomenduotas sekantis žingsnis buvo patikrinti kategorijų balansą. Duomenų pasiskirstymas yra maždaug 83% ir 17%. Akivaizdu, kad egzistuoja disbalansas, tačiau jis nėra kritiškai blogas modeliui. Jeigu disbalansas buvo didesni, derėtų iš naujo sudaryti duomenų imtį ir pakartoti modelio sudarymo procesą. Šiuo atveju galima tęsti su turima imtimi.

Deja, imties balanso savarankiškai patikrinti nesugalvojau.

```
df_logreg['left'].value_counts(normalize=True)
```

```
0    0.831468
1    0.168532
Name: left, dtype: float64
```

Sekančiame žingsnyje apskaičiavau modelio prognozių tikslumo statistikas. Logistinės regresijos modelio precision buvo 79 %, recall - 82 %, f1 rodiklis - 80 % (visi svertiniai vidurkiai). Statistikos indikuoja visai neblogus rezultatus, tačiau jei svarbiausia prognozuoti iš darbo išeinančius darbuotojus, tuomet įverčiai yra gerokai mažesni.

```
target_names = ['Predicted would not leave', 'Predicted would leave']
print(classification_report(y_test, y_pred, target_names=target_names))
```

```
                           precision    recall  f1-score   support

Predicted would not leave       0.86      0.93      0.90      2321
    Predicted would leave       0.44      0.26      0.33       471

                 accuracy                           0.82      2792
                macro avg       0.65      0.60      0.61      2792
             weighted avg       0.79      0.82      0.80      2792
```

Išanalizavęs turimus duomenis ir sudaręs modelį, šiai įmonei galėčiau pateikti šias rekomendacijas:

- Apribokite projektų, su kuriais gali dirbti darbuotojai, skaičių.
- Apsvarstykite galimybę paaukštinti darbuotojus, kurie įmonėje dirba bent ketverius metus, arba atlikite papildomą tyrimą, kodėl ketverius metus dirbantys darbuotojai yra tokie nepatenkinti.
- Apdovanokite darbuotojus už ilgesnį darbo laiką arba nereikalaukite, kad jie dirbtų ilgiau

Refleksija

Šio projekto metu daug išmokau, jaučiu, kad galėčiau bent dalį įgytų žinių pritaikyti realioje darbo situacijoje. Darbo eigoje supratau, jog klaidos bei barjerai, reikalaujantis daugiau laiko nei norėtųsi, yra neišvengiami. Duomenų analitika yra neabejotinai sudėtingas darbas, kurio metu geriausiais draugais tampa Google naršyklė, užrašai ir anksčiau parašytas kodas. Projektas ir Google Advanced Data Analytics kursas padėjo suprasti su kokiais sunkumais gali susidurti su duomenimis dirbantis žmogus. Norint tapti tikrai naudingu profesionalu, ši profesija reikalauja nuolatinio dėmesio, naujų žinių įgijimo, atviro tipo mąstymo. Jaučiuosi patenkintas savimi, kad pavyko įgyvendinti ši projektą bei įveikti kursą.