Predictive Data Analysis

June 20, 2021

1 ANZ Virtual Internship Report

Task 1 Predictive Data Analysis

2 Importing Libraries and Loading Dataset

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor
        from sklearn.linear_model import LinearRegression

In [2]: df = pd.read_excel('C:/Users/hp/Desktop/Data Science-Data@ANZ/Task 1/ANZ synthesised t

In [3]: df.head()
```

Out[3]:

	status	card_present_flag	bpay_biller_code	account	currency	long_lat	txn_description	merchant_id	merchant_code	first_name	...	age	merchant_suburb	merchant_state
0	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	POS	81c48296-73be-44a7-befa-d053f48ce7cd	NaN	Diana	...	26	Ashmore	QLD
1	authorized	0.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	830a451c-316e-4a6a-bf25-e37caedca49e	NaN	Diana	...	26	Sydney	NSW
2	authorized	1.0	NaN	ACC-1222300524	AUD	151.23 -33.94	POS	835c231d-8cdf-4e96-859d-e9d571760cf0	NaN	Michael	...	38	Sydney	NSW
3	authorized	1.0	NaN	ACC-1037050564	AUD	153.10 -27.66	SALES-POS	48514682-c78a-4a88-b0da-2d6302e64673	NaN	Rhonda	...	40	Buderim	QLD
4	authorized	1.0	NaN	ACC-1598451071	AUD	153.41 -27.95	SALES-POS	b4e02c10-0852-4273-b8fd-7b3395e32eb0	NaN	Diana	...	26	Mermaid Beach	QLD

```
                        extraction amount                    transaction_id  \
0  2018-08-01T01:01:15.000+0000  16.25  a623070bfead4541a6b0fff8a09e706c
1  2018-08-01T01:13:45.000+0000  14.19  13270a2a902145da9db4c951e04b51b9
2  2018-08-01T01:26:15.000+0000   6.42  feb79e7ecd7048a5a36ec889d1a94270
3  2018-08-01T01:38:45.000+0000  40.90  2698170da3704fd981b15e64a006079e
4  2018-08-01T01:51:15.000+0000   3.25  329adf79878c4cf0aeb4188b4691c266

     country     customer_id merchant_long_lat movement
0  Australia  CUS-2487424745     153.38 -27.99    debit
1  Australia  CUS-2487424745     151.21 -33.87    debit
2  Australia  CUS-2142601169     151.21 -33.87    debit
3  Australia  CUS-1614226872     153.05 -26.68    debit
4  Australia  CUS-2487424745     153.44 -28.06    debit

[5 rows x 23 columns]
```

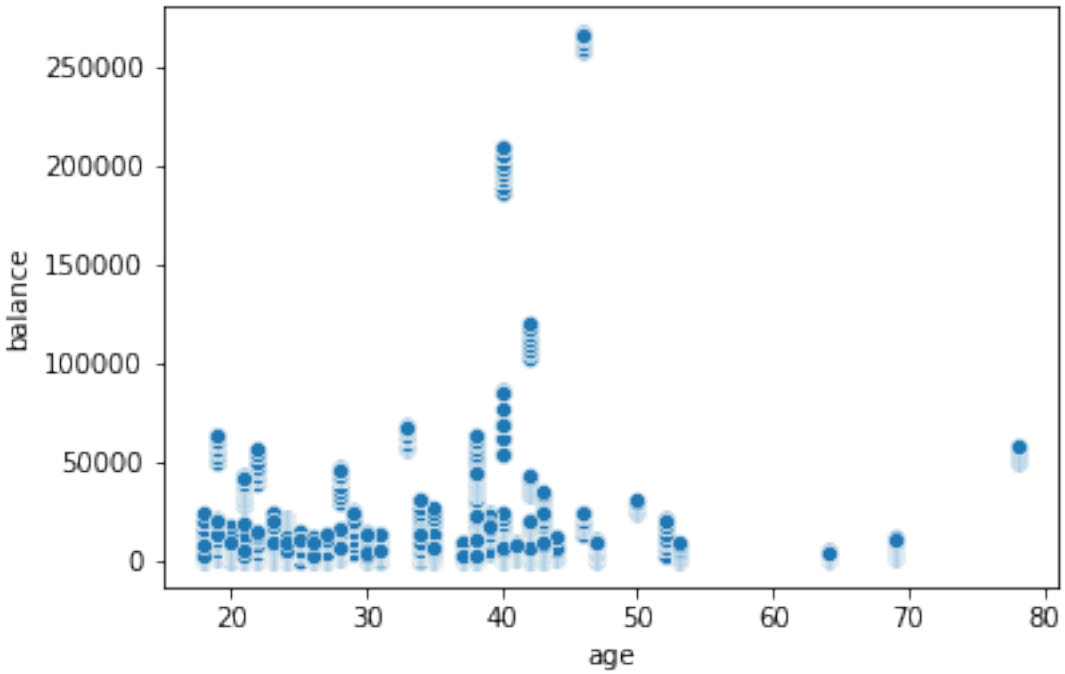
In [4]:
```
data = df[['age', 'amount', 'balance']]
```

In [5]:
```
data.head()
```

Out[5]:
```
   age  amount  balance
0   26   16.25    35.39
1   26   14.19    21.20
2   38    6.42     5.71
3   40   40.90  2117.22
4   26    3.25    17.95
```

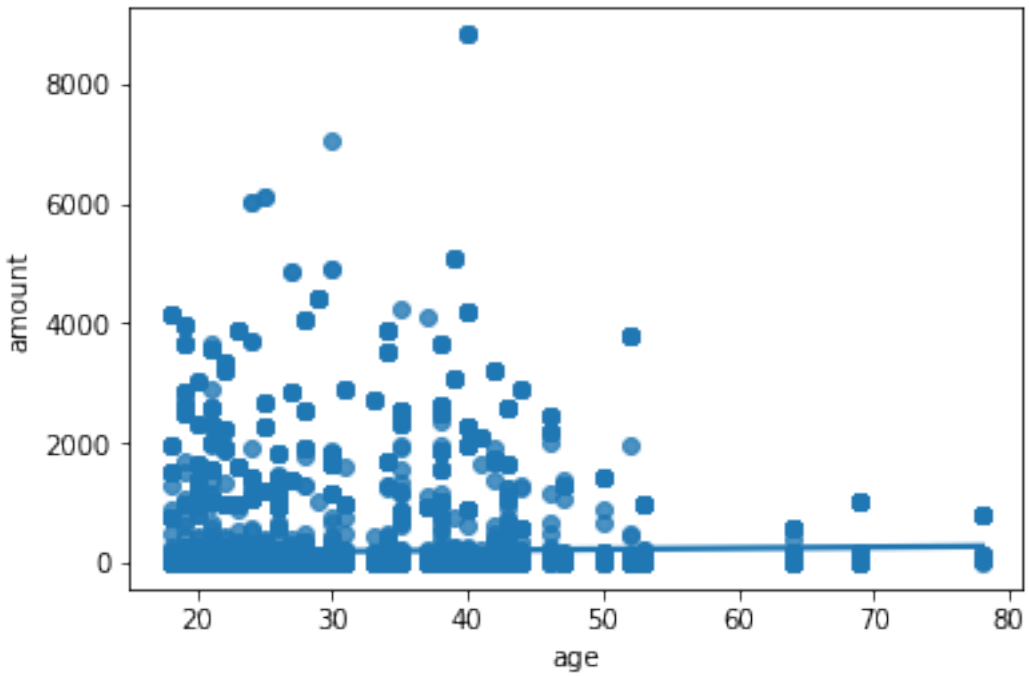
In [6]:
```
sns.scatterplot(x=data['age'], y=data['balance'], data= data)
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa650566d8>
```



In [7]:
```
sns.regplot(x=data['age'], y = data['amount'], data=data)
```

Out[7]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1aa66fc81d0>
```



3 Modifying data to obtain salaries for each customer

In [8]:
```
df_salaries = df[df["txn_description"]=="PAY/SALARY"].groupby("customer_id").mean()
df_salaries.head()
```

Out[8]:

customer_id	card_present_flag	merchant_code	balance	age	amount
CUS-1005756958	NaN	0.0	4718.665385	53	970.47
CUS-1117979751	NaN	0.0	11957.202857	21	3578.65
CUS-1140341822	NaN	0.0	5841.720000	28	1916.51
CUS-1147642491	NaN	0.0	8813.467692	34	1711.39
CUS-1196156254	NaN	0.0	23845.717143	34	3903.73

In [9]:
```
salaries = []

for customer_id in df["customer_id"]:
    salaries.append(int(df_salaries.loc[customer_id]["amount"]))

df["annual_salary"] = salaries
```

```
In [10]: df_cus = df.groupby("customer_id").mean()
         df_cus.head()
```

Out[10]:

customer_id	card_present_flag	merchant_code	balance	age	amount	annual_salary
CUS-1005756958	0.812500	0.0	2275.852055	53	222.862603	970
CUS-1117979751	0.826923	0.0	9829.929000	21	339.843700	3578
CUS-1140341822	0.815385	0.0	5699.212250	28	212.632500	1916
CUS-1147642491	0.750000	0.0	9032.841186	34	245.600169	1711
CUS-1196156254	0.785276	0.0	22272.433755	34	147.145796	3903

4 Predictive Analysis

Linear Regression Model

```
In [11]: N_train = int(len(df_cus)*0.8)
         X_train = df_cus.drop("annual_salary", axis=1).iloc[:N_train]
         Y_train = df_cus["annual_salary"].iloc[:N_train]
         X_test = df_cus.drop("annual_salary", axis=1).iloc[N_train:]
         Y_test = df_cus["annual_salary"].iloc[N_train:]

In [12]: linear_reg = LinearRegression()

In [13]: linear_reg.fit(X_train, Y_train)
         linear_reg.score(X_train, Y_train)

Out[13]: 0.23295376366257825

In [14]: linear_reg.predict(X_test)

Out[14]: array([1993.98473311, 2867.39066481, 1944.95959591, 1806.85984885,
                2226.35045442, 2075.34697175, 1813.02987337, 5388.67435983,
                1902.35351608, 2191.90445145, 1713.48134178, 2854.40519949,
                2094.77781158, 3815.34342881, 2249.92922822, 1768.80816189,
                2095.02988288, 1515.18425875, 1782.72752537, 2481.2898546 ])

In [15]: linear_reg.score(X_test, Y_test)

Out[15]: -0.3169423498074737
```

Decision Tree - Classification and Regression

```
In [16]: df_cat = df[["txn_description", "gender", "age", "merchant_state", "movement"]]

In [17]: pd.get_dummies(df_cat).head()
```

Out[17]:

	age	txn_description_INTER BANK	txn_description_PAY/SALARY	txn_description_PAYMENT	txn_description_PHONE BANK	txn_description_POS	txn_description_SALES-POS	gender_F	gender_M	merchant_state_ACT	merchant_state_NSW	merchant_state_NT	merchant_state_QLD	merchant_state_SA	merchant_state_TAS	merchant_state_VIC	merchant_state_WA	movement_credit	movement_debit
0	26	0	0	0	0	1	0	1	0	0	0	0	1	0	0	0	0	0	1
1	26	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	1
2	38	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0	0	0	1
3	40	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1
4	26	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0	0	0	1

```
In [18]: N_train = int(len(df)*0.8)
         X_train = pd.get_dummies(df_cat).iloc[:N_train]
```

```
Y_train = df["annual_salary"].iloc[:N_train]
X_test = pd.get_dummies(df_cat).iloc[N_train:]
Y_test = df["annual_salary"].iloc[N_train:]
```

Classification

```
In [19]: decision_tree_class = DecisionTreeClassifier()

In [20]: decision_tree_class.fit(X_train, Y_train)
         decision_tree_class.score(X_train, Y_train)

Out[20]: 0.7882499481004774

In [21]: decision_tree_class.predict(X_test)

Out[21]: array([1013, 1043, 4132, ..., 4054, 1043,  996], dtype=int64)
```

Regression

```
In [22]: decision_tree_reg = DecisionTreeRegressor()

In [23]: decision_tree_reg.fit(X_train, Y_train)
         decision_tree_reg.score(X_train, Y_train)

Out[23]: 0.7468978726536879

In [24]: decision_tree_reg.predict(X_test)

Out[24]: array([1226.42857143, 1043.        , 4132.        , ..., 3345.04761905,
                1043.        , 1626.        ])

In [25]: decision_tree_reg.score(X_test, Y_test)

Out[25]: 0.6799861150839813
```