Data@Anz- Data-Analysis

August 18, 2021

0.0.1 1.0 Import Libraries

[12]:
```
#remove warnings
import warnings
warnings.filterwarnings('ignore')

# Data analysis and wrangling
import pandas as pd
import numpy as np
import statistics
# for data visualization
import seaborn as sns
%matplotlib inline
from matplotlib import pyplot as plt
from matplotlib import style
get_ipython().run_line_magic('matplotlib', 'inline')
# For Dates Conversion
import datetime
# for selection of Algorithms
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn import linear_model
from sklearn.linear_model import Perceptron
from sklearn.linear_model import SGDClassifier
from sklearn.model_selection import train_test_split
```

[13]:
```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import cross_val_score
# Import more libraries
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_matrix,␣
↪classification_report
```

```python
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from statsmodels.tools.eval_measures import rmse
from sklearn import linear_model
from sklearn import metrics
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error as MSE
pd.set_option('max_rows', None)
```

0.0.2 2.0 Getting and Loading Dataset

[16]:
```python
#Getting the dataset
import os

# Path of the file to read

Anz_dataset = pd.read_csv(r"C:\Users\User\Downloads\DATA SCIENCE\anz.csv")
```

0.0.3 3.0 Exploratory Analysis

[17]:
```python
#checking columns
Anz_dataset.columns
```

[17]:
```
Index(['status', 'card_present_flag', 'bpay_biller_code', 'account',
       'currency', 'long_lat', 'txn_description', 'merchant_id',
       'merchant_code', 'first_name', 'balance', 'date', 'gender', 'age',
       'merchant_suburb', 'merchant_state', 'extraction', 'amount',
       'transaction_id', 'country', 'customer_id', 'merchant_long_lat',
       'movement'],
      dtype='object')
```

[18]:
```python
#checking info about the Anz_dataset
Anz_dataset.head()

print(Anz_dataset.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12043 entries, 0 to 12042
Data columns (total 23 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   status             12043 non-null  object
 1   card_present_flag  7717 non-null   float64
 2   bpay_biller_code   885 non-null    object
 3   account            12043 non-null  object
 4   currency           12043 non-null  object
 5   long_lat           12043 non-null  object
```

```
 6   txn_description     12043 non-null  object
 7   merchant_id         7717 non-null   object
 8   merchant_code       883 non-null    float64
 9   first_name          12043 non-null  object
 10  balance             12043 non-null  float64
 11  date                12043 non-null  object
 12  gender              12043 non-null  object
 13  age                 12043 non-null  int64
 14  merchant_suburb     7717 non-null   object
 15  merchant_state      7717 non-null   object
 16  extraction          12043 non-null  object
 17  amount              12043 non-null  float64
 18  transaction_id      12043 non-null  object
 19  country             12043 non-null  object
 20  customer_id         12043 non-null  object
 21  merchant_long_lat   7717 non-null   object
 22  movement            12043 non-null  object
dtypes: float64(4), int64(1), object(18)
memory usage: 2.1+ MB
None
```

There missing values in some columns

[19]:
```
#printing data shape
print('Anz_dataset shape: ',Anz_dataset.shape)
#printing the number of rows
print('Number  of  rows: ',len(Anz_dataset))
```

```
Anz_dataset shape:  (12043, 23)
Number  of  rows:  12043
```

[20]:
```
# checking statisics summary of Anz_dataset
Anz_dataset.describe()
```

[20]:
```
       card_present_flag  merchant_code        balance           age  \
count        7717.000000          883.0   12043.000000  12043.000000
mean            0.802644            0.0   14704.195553     30.582330
std             0.398029            0.0   31503.722652     10.046343
min             0.000000            0.0       0.240000     18.000000
25%             1.000000            0.0    3158.585000     22.000000
50%             1.000000            0.0    6432.010000     28.000000
75%             1.000000            0.0   12465.945000     38.000000
max             1.000000            0.0  267128.520000     78.000000

             amount
count  12043.000000
mean     187.933588
std      592.599934
```

```
min         0.100000
25%        16.000000
50%        29.000000
75%        53.655000
max      8835.980000
```

Observation:

There is inconsistency and lots of zero values in merchant code column.It's probably a categorical column.

4.1 Checking for unique customers

[21]:
```
# Checking for the 100 unique customers
print("Number of unique customer ID's: ", Anz_dataset.customer_id.
 ↪nunique())
print("Number of unique transaction ID's: ", Anz_dataset.transaction_id.
 ↪nunique())
print('Number of unique accounts: ', Anz_dataset.account.nunique())
```

```
Number of unique customer ID's:  100
Number of unique transaction ID's: 12043
Number of unique accounts:  100
```

4.2 Checking the format of Date

[22]:
```
Anz_dataset.date.describe()
#Anz_dataset.date.count()
```

[22]:
```
count          12043
unique            91
top        9/28/2018
freq             174
Name: date, dtype: object
```

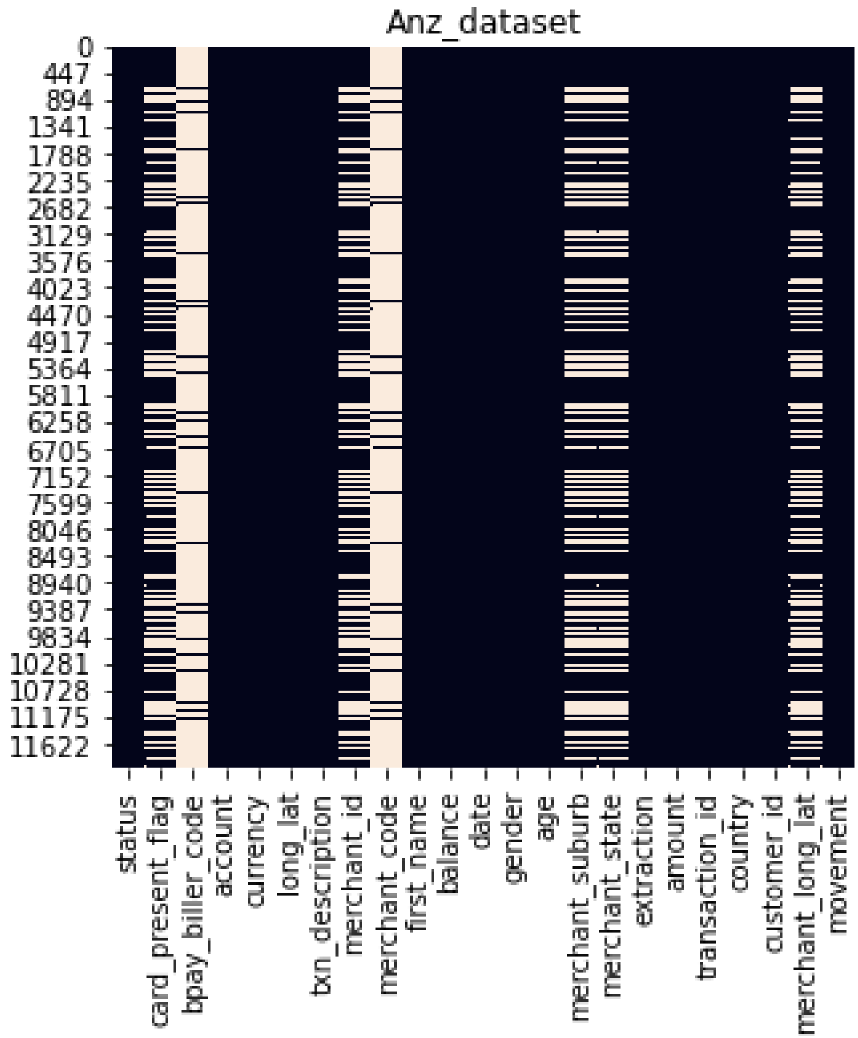
Observation:

One day is missing. Date format is consistent

0.0.4 5.0 Missing Values

[23]:
```
#Checking for missing values

def missing_data(Anz_dataset, title):
    fig, ax = plt.subplots(figsize=(5,5))
    plt.title(title)
    sns.heatmap(Anz_dataset.isnull(), cbar=False)
```

[24]:
```
missing_data(Anz_dataset, "Anz_dataset")
```



Observantion:

The missing columns are evident from the graph

0.0.5 5.1 Checking Percentage of missing values

[25]:
```
# checking percentage of missing values
missing_value = Anz_dataset.isnull().sum()
missing_value = missing_value[missing_value > 0]
percentage_missing_value = round(missing_value / len(Anz_dataset), 3) * 100

pd.DataFrame({"Number  of missing_value": missing_value,"Percentage":␣
↪percentage_missing_value}).sort_values(by = "Percentage", ascending = False)
```

[25]:

	Number of missing_value	Percentage
bpay_biller_code	11158	92.7
merchant_code	11160	92.7
card_present_flag	4326	35.9
merchant_id	4326	35.9
merchant_suburb	4326	35.9
merchant_state	4326	35.9
merchant_long_lat	4326	35.9

Observation:

merchant_code and bpay_biller_code have a high percentage of missing values.We are dropping the two columns due to high percentage of missing values.

0.0.6 5.2 Treating Null values

[26]:
```
Anz_dataset.drop(columns=[ 'merchant_code','bpay_biller_code'],inplace=True)
```

[27]:
```
#dropping null values in merchant
df = pd.DataFrame(Anz_dataset.merchant_state)
new_df = df.dropna()
new_df.head()
```

[27]:

	merchant_state
0	QLD
1	NSW
2	NSW
3	QLD
4	QLD

Confirming the drop

[28]:
```
missing_value = new_df.isnull().sum()
print(missing_value)
#Anz_dataset.head(2)
```

```
merchant_state    0
dtype: int64
```

[29]:
```
Anz_dataset.columns
```

[29]:
```
Index(['status', 'card_present_flag', 'account', 'currency', 'long_lat',
       'txn_description', 'merchant_id', 'first_name', 'balance', 'date',
       'gender', 'age', 'merchant_suburb', 'merchant_state', 'extraction',
       'amount', 'transaction_id', 'country', 'customer_id',
       'merchant_long_lat', 'movement'],
      dtype='object')
```

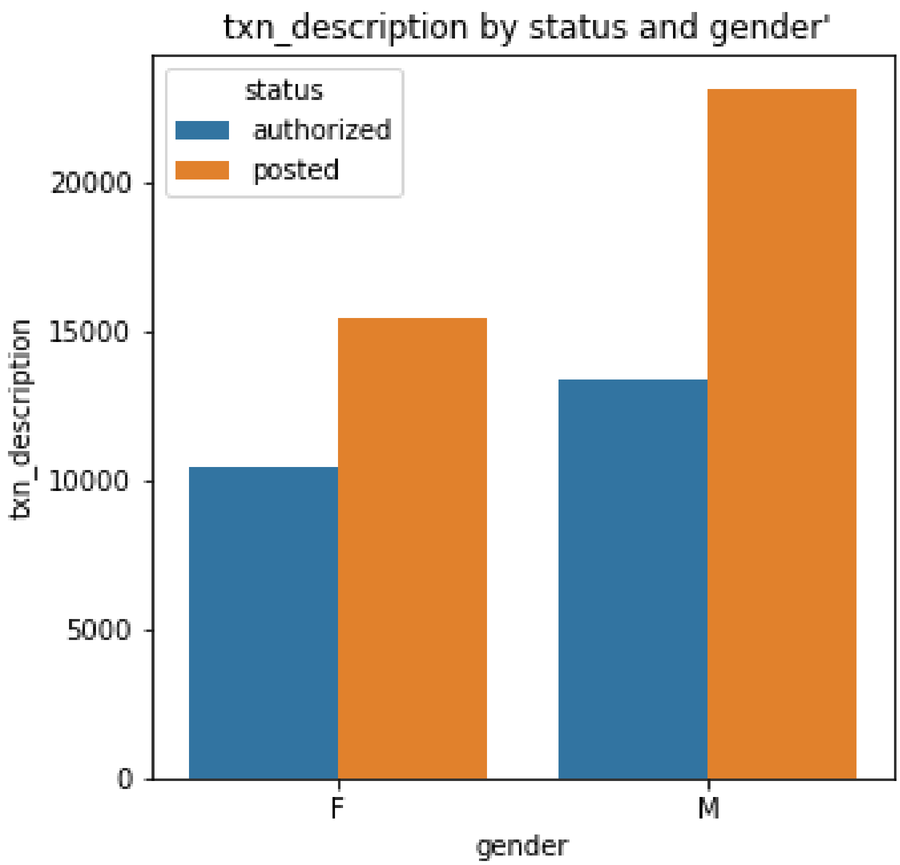
0.1 6.0 Analysis of Features

Distribution plot function

0.1.1 Analysis of gender and status

[30]:
```
def bar_chart_compare(dataset, feature1, feature2=None, title = "'
↪txn_description by status and gender'"):
    plt.figure(figsize = [5,5])
    plt.title(title)
    g = sns.barplot(x=feature1, y='balance', hue=feature2, ci=None,
↪data=dataset).set_ylabel('txn_description')
```

[31]:
```
bar_chart_compare(Anz_dataset, "gender", "status")
```

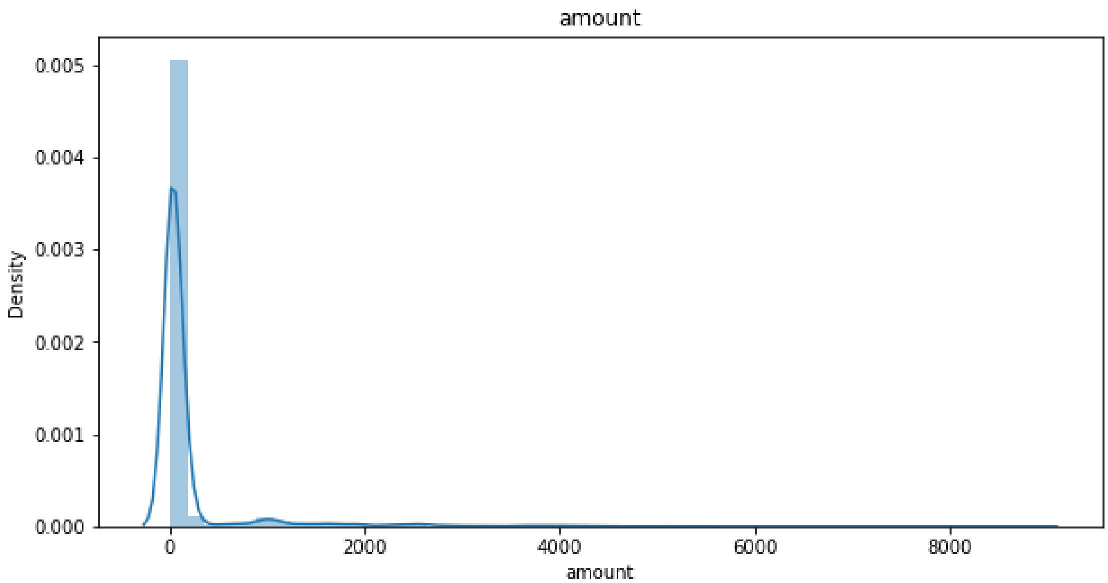


There were more males authorized and posted than females

0.1.2 Gender and Amount

[32]:
```
plt.figure(figsize = (10,5))
sns.distplot(Anz_dataset.amount)
plt.title("amount")
```

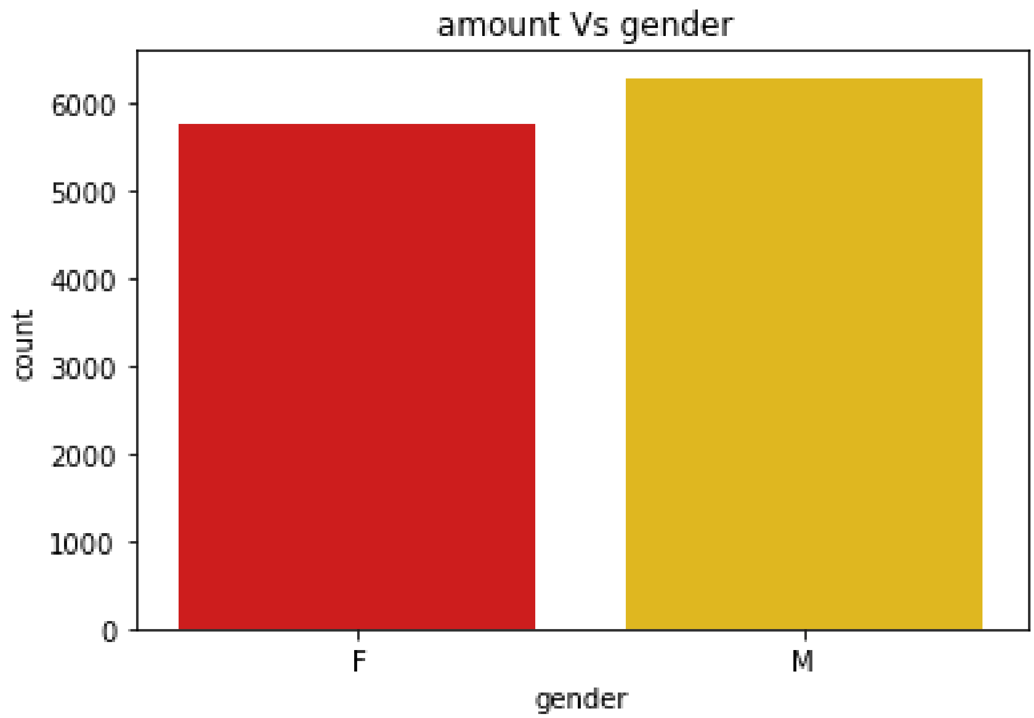
[32]: Text(0.5, 1.0, 'amount')



amount distribution is right or positively skewed

[33]:
```
sns.countplot(x= "gender",data = Anz_dataset, palette ="hot")
plt.title(" amount Vs gender")
```

[33]: Text(0.5, 1.0, ' amount Vs gender')



Observation:

More males engaged in transation than females. the dsitribution

Transaction

[34]:
```
# checking Average
tran_average = statistics.mean(Anz_dataset.amount)
print('average transactional amount is:',  round(tran_average, 2))
```

```
average transactional amount is: 187.93
```

[35]:
```
Anz_dataset.groupby('txn_description').amount.mean()
```

[35]:
```
txn_description
INTER BANK       86.699461
PAY/SALARY     1898.728029
PAYMENT          77.613077
PHONE BANK      106.099010
POS              40.407412
SALES-POS        39.909789
Name: amount, dtype: float64
```

[36]:
```
# Statistical summary
Anz_dataset.groupby('txn_description').amount.describe()
```

[36]:

```
                   count         mean          std    min       25%       50%  \
txn_description
INTER BANK         742.0    86.699461   198.706044   16.0    26.000    39.000
PAY/SALARY         883.0  1898.728029  1150.364621  576.0  1013.670  1626.480
PAYMENT           2600.0    77.613077   152.310315   15.0    32.000    42.500
PHONE BANK         101.0   106.099010   245.999695   21.0    36.000    43.000
POS               3783.0    40.407412   165.771678    0.1    12.035    19.430
SALES-POS         3934.0    39.909789   132.734185    0.1    12.160    20.035

                      75%      max
txn_description
INTER BANK         83.000  1956.00
PAY/SALARY       2538.680  8835.98
PAYMENT            70.000  1981.00
PHONE BANK         67.000  1916.00
POS                33.155  7081.09
SALES-POS          34.575  4233.00
```

Observation:

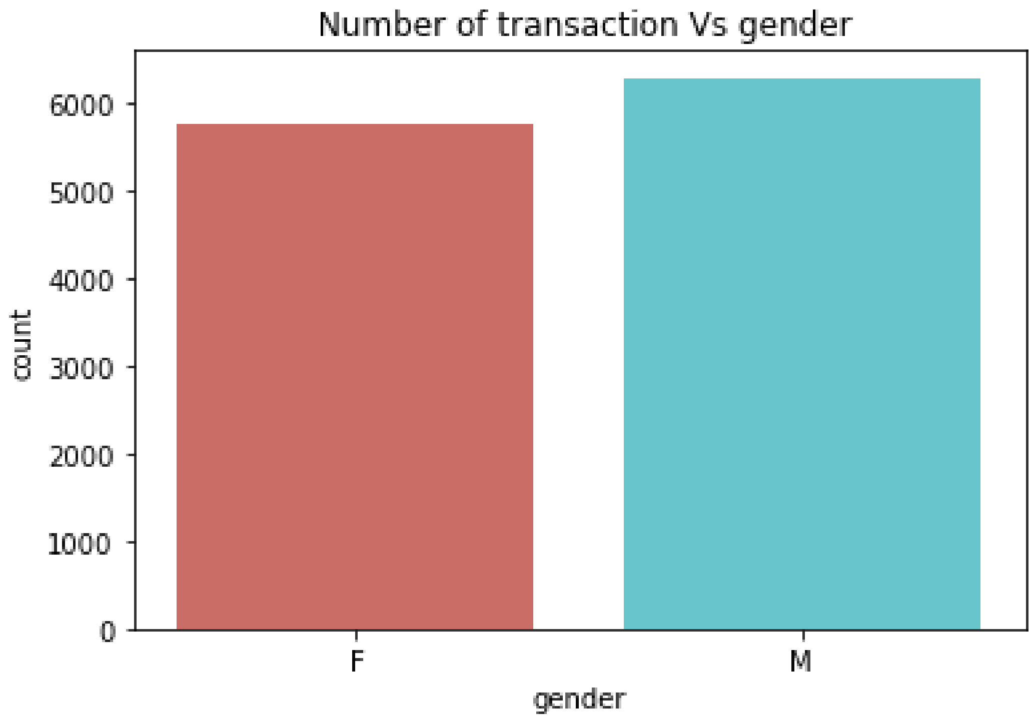
There is a huge difference between the minimum and the maximum numbers. There is high confidence interval, it means the sample mean is not reliable.

0.1.3 Analysizing Features gender and Number of transaction

[37]:

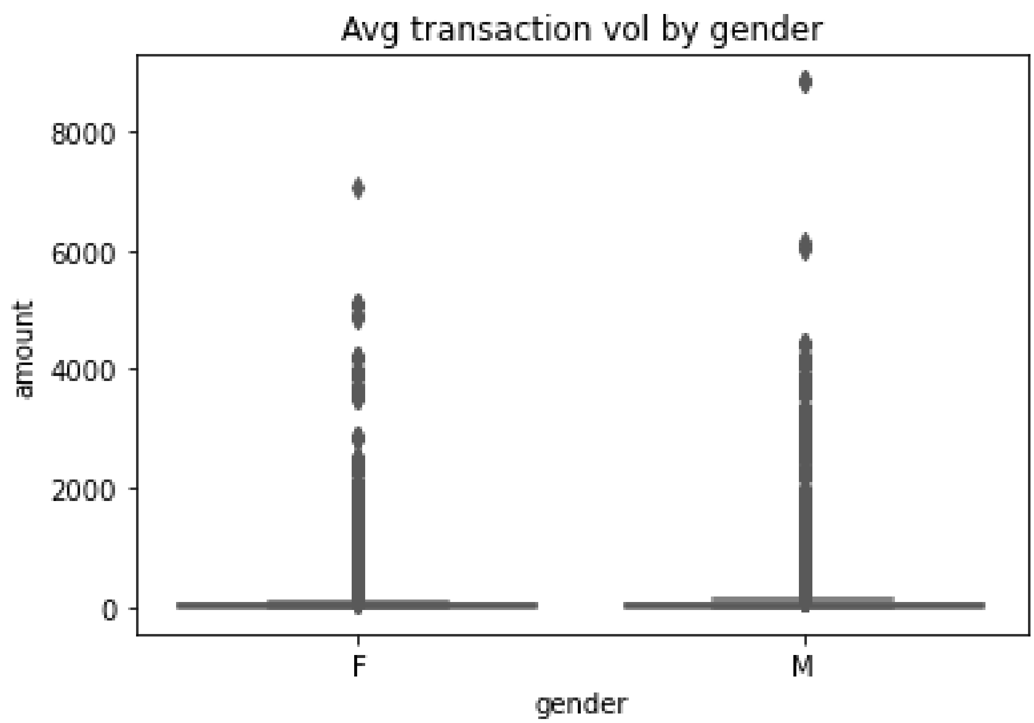
```
sns.countplot(x= "gender",data = Anz_dataset,  palette ="hls")
plt.title(" Number of transaction Vs gender")
```

[37]: Text(0.5, 1.0, ' Number of transaction Vs gender')



[38]:
```
sns.boxplot(x= 'gender', y= 'amount', data = Anz_dataset, palette = "Set2")
plt.title('Avg transaction vol by gender')
```

[38]: Text(0.5, 1.0, 'Avg transaction vol by gender')



There are more male customers with transactions than females This is a confirmed case.

[39]:
```
#Average transaction volume by state and movement

Anz_dataset.merchant_suburb.dropna().head()
```

[39]:
```
0            Ashmore
1             Sydney
2             Sydney
3            Buderim
4      Mermaid Beach
Name: merchant_suburb, dtype: object
```

[40]:
```
Anz_dataset.card_present_flag.dropna().isnull().sum()
```

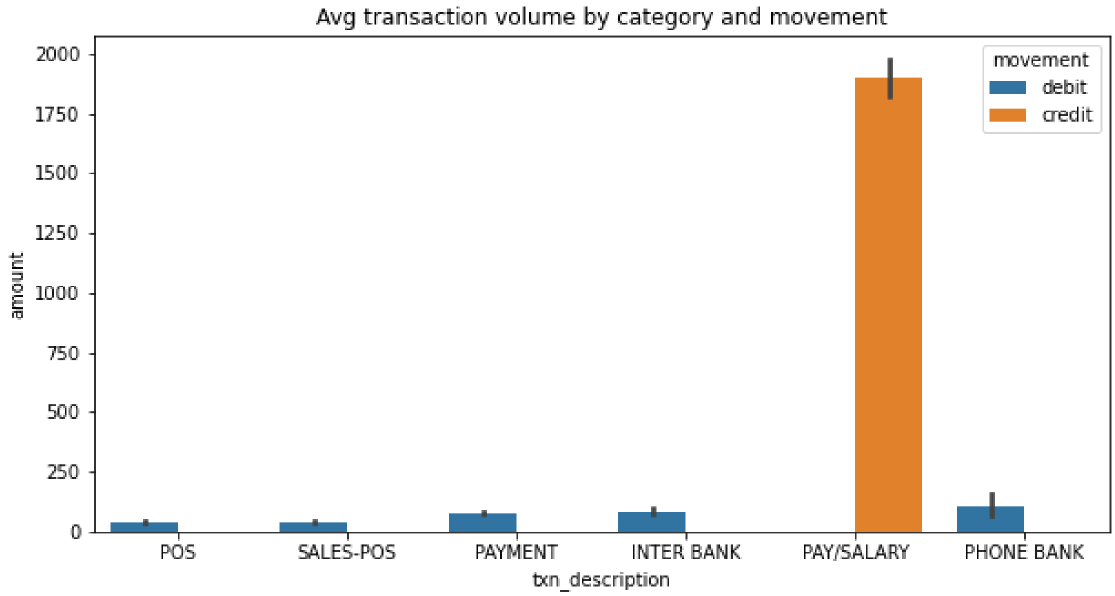
[40]:
```
0
```

Observation:

PaySalary is the highest transaction, the margin is wide.

[41]:
```
plt.figure(figsize = (10,5))
sns.barplot(x='txn_description',y= 'amount',data = Anz_dataset,␣
↪hue='movement',palette ="tab10")
plt.title('Avg transaction volume by category and movement')
```

[41]: Text(0.5, 1.0, 'Avg transaction volume by category and movement')

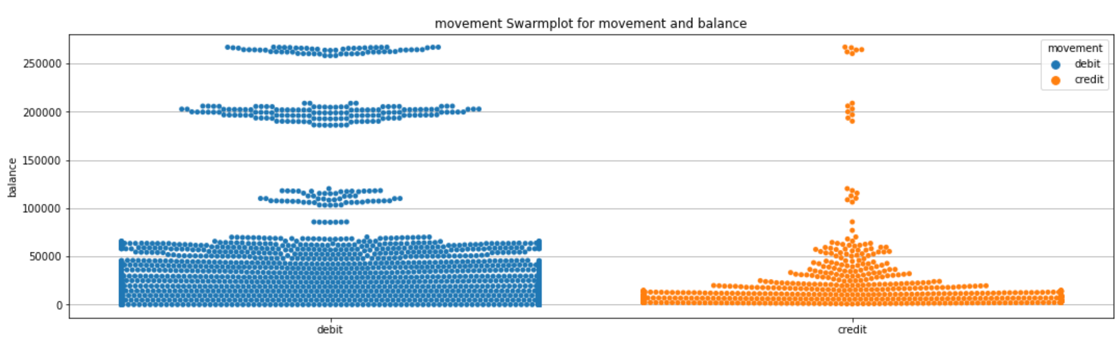


Observations:

salaries have the highest transaction and were paid with credit. others used debit and no credit. There is high confidence interval,implying the sample mean was not reliable as an estimate of the true amount of the salary,Interbank and Phone bank. This means the average portrayed is false.

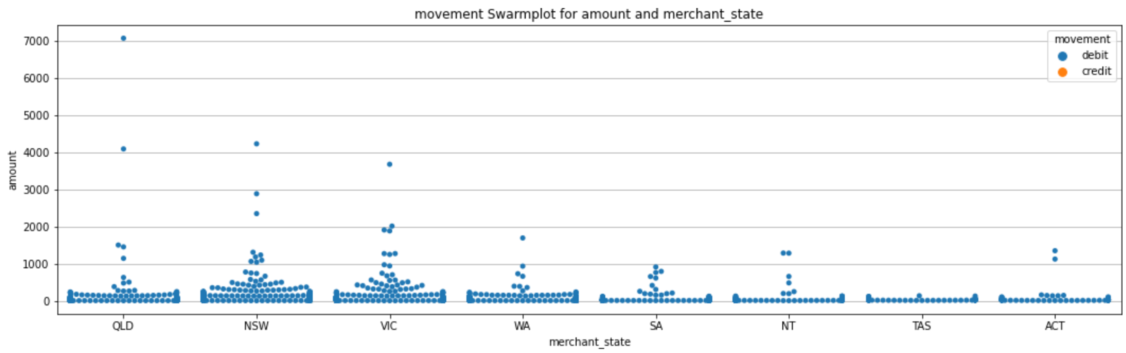
[42]:
```
def plot_swarm_movement(dataset, feature1, feature2, title, fize = (155)):
    fig, ax = plt.subplots(figsize=(18,5))
    # Turns off grid on the left Axis.
    ax.grid(True)
    plt.xticks(list(range(0,100,2)))
    sns.swarmplot(y=feature1, x=feature2, hue='movement',data=Anz_dataset).
 ↪set_title(title)
```

[43]:
```
plot_swarm_movement(Anz_dataset,'balance','movement','movement Swarmplot for␣
 ↪movement and balance')
```



From the swarm plot the debit is performing well than the credit.

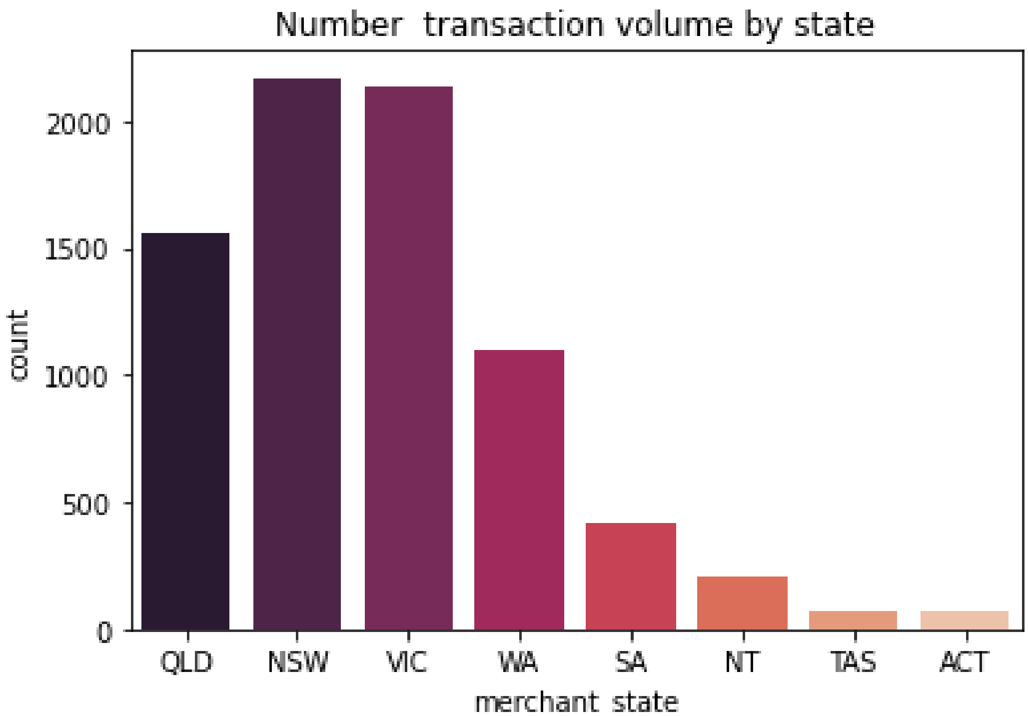
[44]:
```
plot_swarm_movement(Anz_dataset,'amount','merchant_state','movement Swarmplot␣
↪for amount and merchant_state')
```



ACT's average transaction volume is the highest but it is the state with the lowest number of transaction. This means that, the company needs to focus on ACT since its average tranasaction is high. While the NSW and VIC have a high number of transactions, their average transactions volume is relatively low. Hence, little effort should be put there.

[45]:
```
sns.countplot(x='merchant_state',data = Anz_dataset, palette ="rocket")
plt.title('Number  transaction volume by state')
```

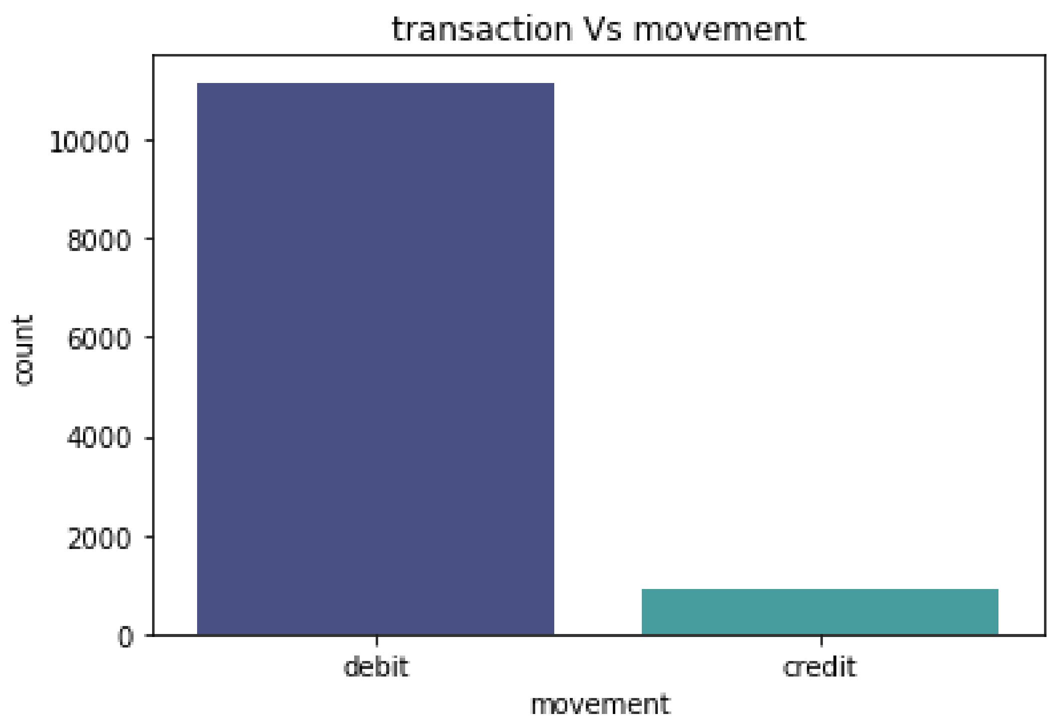
[45]: Text(0.5, 1.0, 'Number transaction volume by state')



Observation: There are more male customers with transactions than females ACT's average transaction volume is the highest but it is the state with the lowest number of transaction

[46]:
```
sns.countplot(x= 'movement',data = Anz_dataset,palette ="mako")
plt.title('transaction Vs movement')
```

[46] : Text(0.5, 1.0, 'transaction Vs movement')



There are more debit transactions than credit

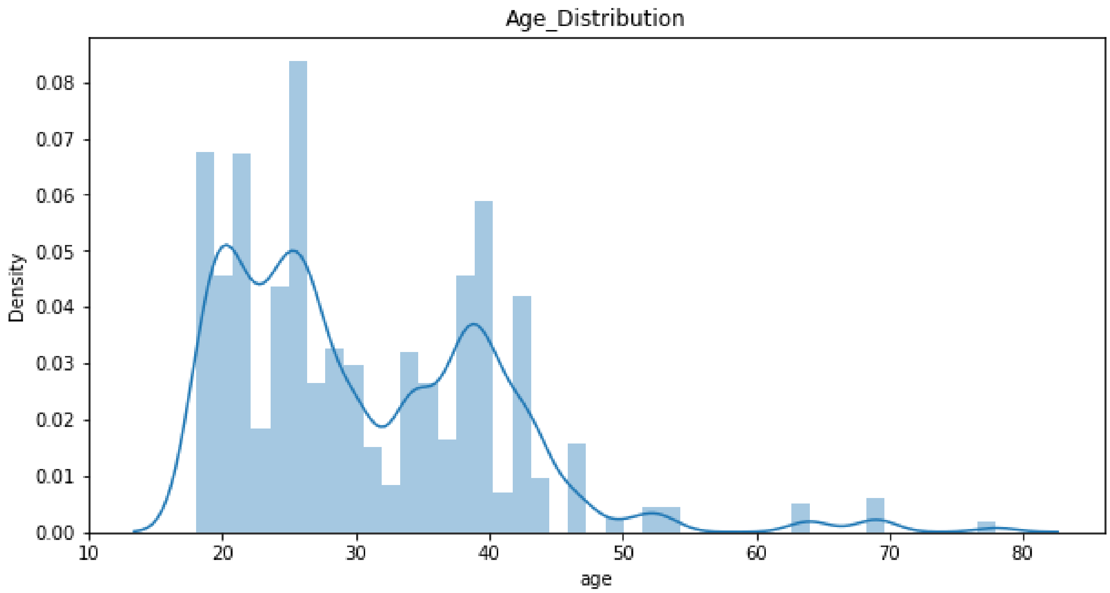
[ ]:
```
plt.figure(figsize = (10,5))
sns.barplot(x='merchant_state', y='amount',data = Anz_dataset, ␣
↪hue='movement', palette ="coolwarm")
plt.title('Average transaction volume by state and movement')
```

Observation: ACT has the highest average transaction volume but the variance is quite large.

0.1.4 Age distribution

[48]:
```
plt.figure(figsize = (10,5))
sns.distplot(Anz_dataset.age)
plt.title("Age_Distribution")
```

[48]: Text(0.5, 1.0, 'Age_Distribution')



lowest transactions came from people after 50 year while most of transactions came from people in 20's.

Transaction Volume

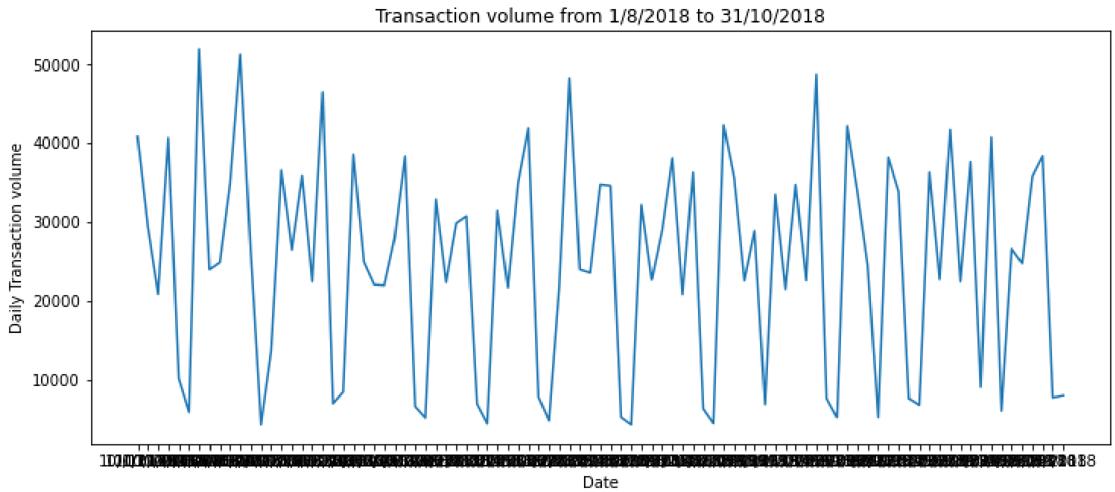
[49]:
```
daily = pd.DataFrame(Anz_dataset.groupby("date").amount.sum())
daily.head()
```

[49] :

date	amount
10/1/2018	40823.03
10/10/2018	29399.50
10/11/2018	20851.67
10/12/2018	40658.20
10/13/2018	10140.81

[50]:
```
fig, ax = plt.subplots(figsize = (12, 5))
ax.plot(daily.index, daily.amount)
plt.title("Transaction volume from 1/8/2018 to 31/10/2018")
plt.xlabel("Date")
plt.ylabel("Daily Transaction volume")
```

[50] : Text(0, 0.5, 'Daily Transaction volume')

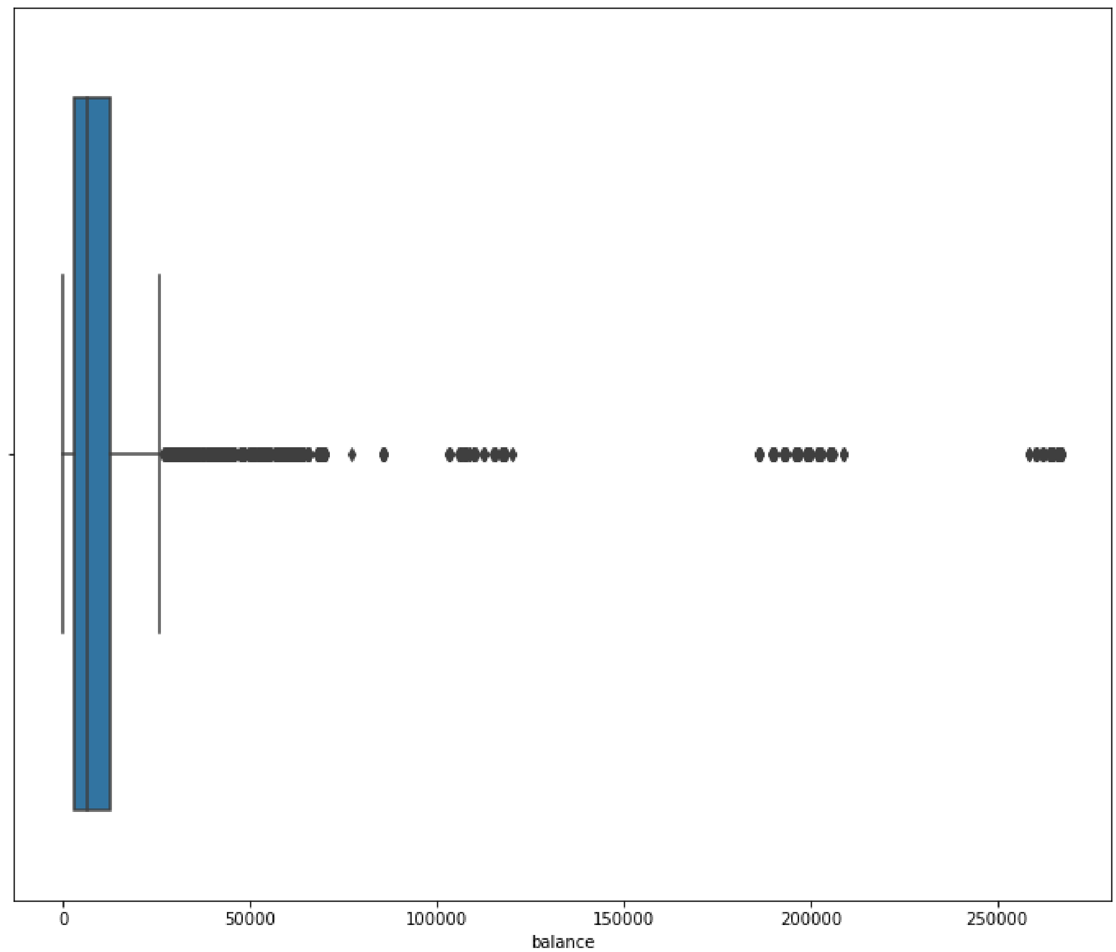


There is a similiar of pattern of rising and dropping transcation volume

0.1.5 Balance Distribution

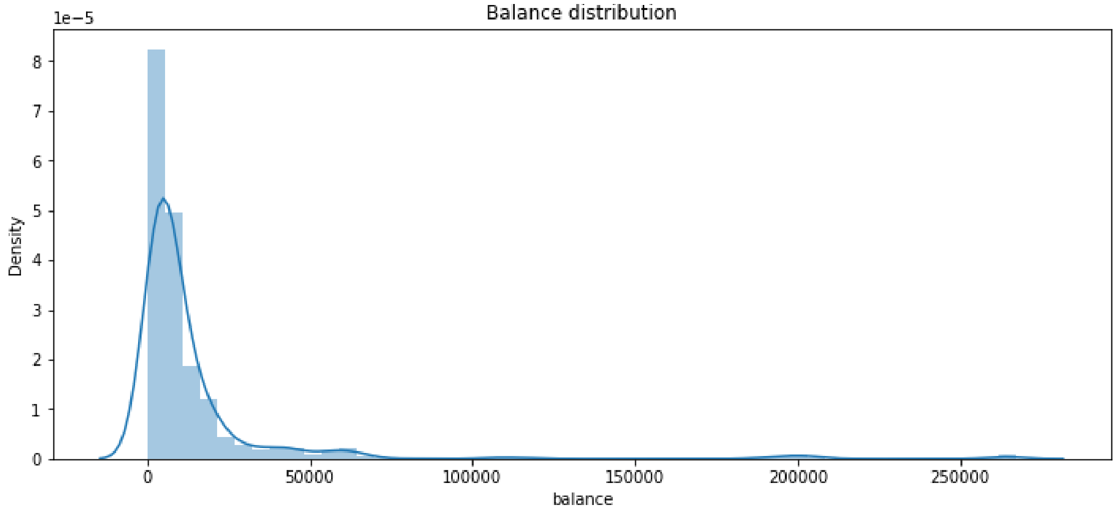
[51]:
```
plt.figure(figsize = (12, 10))
sns.boxplot(Anz_dataset.balance)
```

[51]: <AxesSubplot:xlabel='balance'>



[52]:
```
plt.figure(figsize = (12, 5))
sns.distplot(Anz_dataset.balance)
plt.title("Balance  distribution")
```

[52]: Text(0.5, 1.0, 'Balance distribution')



The balance session is rightly skewed, hence, needs to be corrected before the model.

[53]:
```
customer_monthly_volume  =  pd.DataFrame(Anz_dataset.groupby("customer_id").
↪amount.
sum()/3)
customer_monthly_volume.head()
```

[53]:

	amount
customer_id	
CUS-1005756958	5422.990000
CUS-1117979751	11328.123333
CUS-1140341822	5670.200000
CUS-1147642491	9660.273333
CUS-1196156254	12016.906667

[54]:
```
pd.DataFrame(Anz_dataset.groupby("customer_id").amount.sum())
```

[54]:

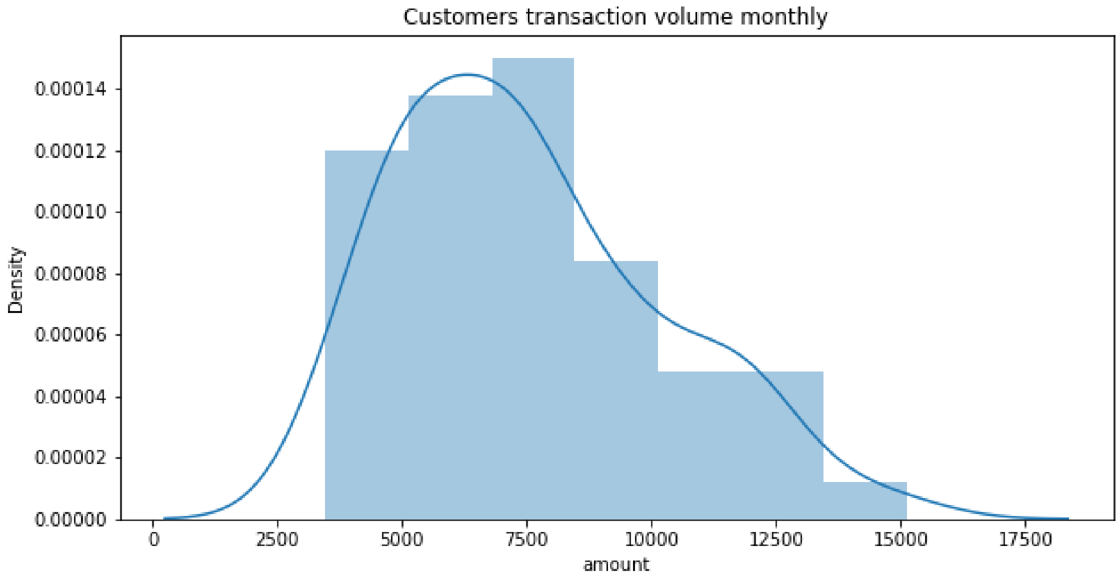
	amount
customer_id	
CUS-1005756958	16268.97
CUS-1117979751	33984.37
CUS-1140341822	17010.60
CUS-1147642491	28980.82
CUS-1196156254	36050.72
CUS-1220154422	20596.11
CUS-1233833708	10385.54
CUS-1271030853	29079.78
CUS-127297539	21856.81
CUS-134193016	17230.81
CUS-134833760	32243.17

CUS-1388323263	18680.22
CUS-1433879684	15563.36
CUS-1462656821	34511.20
CUS-1478398256	27698.18
CUS-1499065773	19609.82
CUS-1604596597	19221.00
CUS-1609060617	23378.97
CUS-1614226872	19286.44
CUS-1617121891	33085.75
CUS-164374203	27722.98
CUS-1646183815	10845.25
CUS-1646621553	22141.51
CUS-1654129794	10587.42
CUS-1669695324	23070.56
CUS-1739931018	10652.72
CUS-1790886359	28489.54
CUS-1816693151	40215.54
CUS-1842679196	12438.05
CUS-1892177589	23222.91
CUS-1896554896	20818.80
CUS-1928710999	22729.12
CUS-2031327464	35832.97
CUS-2059096722	26234.46
CUS-2083971310	21230.42
CUS-2110742437	13645.24
CUS-2142601169	23696.45
CUS-2155701614	37943.79
CUS-2178051368	16033.50
CUS-2206365095	15062.14
CUS-2283904812	16942.84
CUS-2317998716	21422.69
CUS-2348881191	18754.06
CUS-2370108457	20006.47
CUS-2376382098	32198.92
CUS-2484453271	12630.60
CUS-2487424745	26211.59
CUS-2500783281	26408.22
CUS-2505971401	29711.92
CUS-2599279756	14763.14
CUS-261674136	36786.13
CUS-2630892467	13614.83
CUS-2650223890	17635.02
CUS-2663907001	33459.56
CUS-2688605418	20550.24
CUS-2695611575	21479.97
CUS-2738291516	45409.16
CUS-2819545904	28265.48

```
CUS-2977593493  13522.71
CUS-3026014945  29074.34
CUS-3117610635  22820.90
CUS-3129499595  21006.53
CUS-3142625864  42688.30
CUS-3151318058  16424.00
CUS-3174332735  24950.27
CUS-3180318393  22792.27
CUS-3201519139  14535.11
CUS-3249305314  32401.50
CUS-325142416   27483.01
CUS-3255104878  13259.72
CUS-326006476   15687.97
CUS-331942311   14922.51
CUS-3325710106  20515.56
CUS-3336454548  34020.50
CUS-3378712515  22761.96
CUS-3395687666  14216.01
CUS-3431016847  15324.86
CUS-3462882033  22801.59
CUS-3702001629  15463.59
CUS-3716701010  23233.50
CUS-3904958894  14692.24
CUS-3989008654  19160.97
CUS-4023861240  26521.55
CUS-4123612273  22312.12
CUS-4142663097  36588.25
CUS-423725039   13864.12
CUS-443776336   16932.26
CUS-495599312   21502.62
CUS-497688347   15400.63
CUS-511326734   16615.46
CUS-51506836    24100.75
CUS-527400765   36543.61
CUS-537508723   23516.20
CUS-55310383    28367.16
CUS-586638664   17222.50
CUS-72755508    11438.37
CUS-809013380   18810.09
CUS-860700529   18099.88
CUS-880898248   11462.45
CUS-883482547   36639.41
```

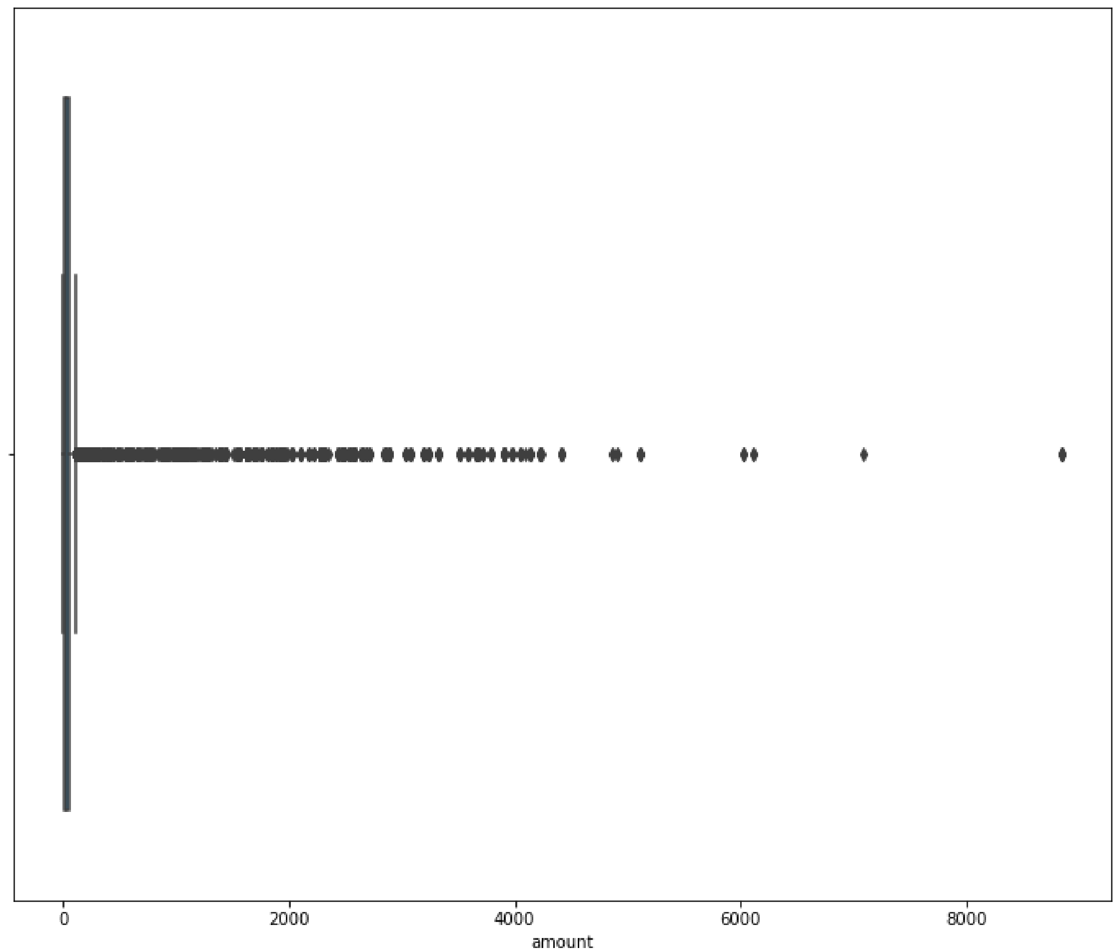
[55]:
```python
plt.figure(figsize = (10, 5))
sns.distplot(customer_monthly_volume.amount)
plt.title("Customers transaction volume monthly")
```

[55] : Text(0.5, 1.0, 'Customers transaction volume monthly')



[56]:
```
plt.figure(figsize = (12, 10))
sns.boxplot(Anz_dataset.amount)
```

[56] : <AxesSubplot:xlabel='amount'>



The amount session is also rightly skewed and hence has to be transformed before the model.

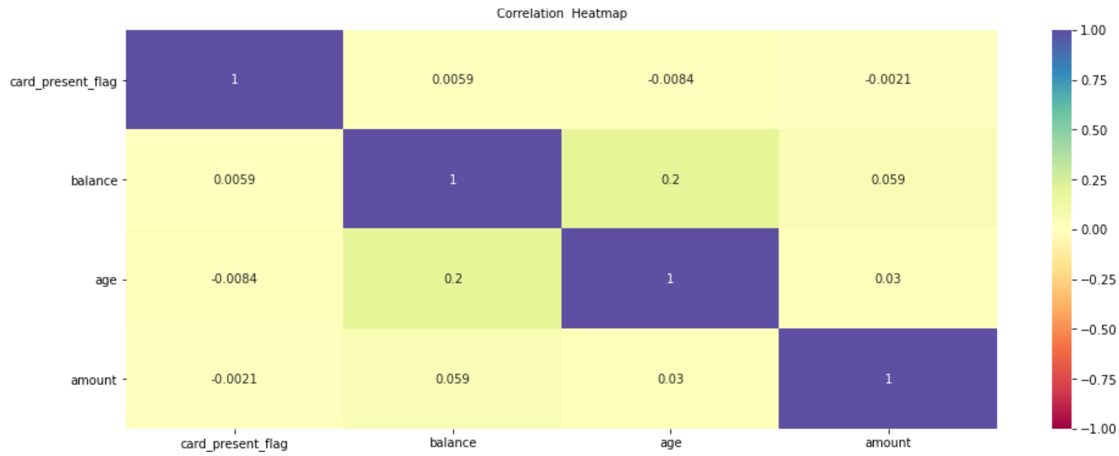
0.1.6 Checking Correlation

[57]:
```
Anz_dataset.corr()
```

[57]:

	card_present_flag	balance	age	amount
card_present_flag	1.000000	0.005925	-0.008405	-0.002074
balance	0.005925	1.000000	0.199329	0.059178
age	-0.008405	0.199329	1.000000	0.029980
amount	-0.002074	0.059178	0.029980	1.000000

[58]:
```
plt.figure(figsize=(16, 6))
heatmap = sns.heatmap(Anz_dataset.corr(), vmin=-1, vmax=1, annot=True, ␣
↪cmap='Spectral')
heatmap.set_title('Correlation Heatmap',fontdict={'fontsize':10}, pad=10);
```



There are correlation in the dataset,We do not need to drop any column.

[59]:
```
Anz_dataset.head(1)
#type(Anz_dataset)
```

[59] :
```
      status  card_present_flag          account currency       long_lat  \
0  authorized                1.0  ACC-1598451071      AUD  153.41 -27.95

  txn_description                           merchant_id first_name  balance  \
0             POS  81c48296-73be-44a7-befa-d053f48ce7cd      Diana    35.39

       date  ... age  merchant_suburb  merchant_state  \
0  8/1/2018  ...  26          Ashmore             QLD

                     extraction  amount                    transaction_id  \
0  2018-08-01T01:01:15.000+0000   16.25  a623070bfead4541a6b0fff8a09e706c

     country      customer_id merchant_long_lat movement
0  Australia  CUS-2487424745     153.38 -27.99    debit

[1 rows x 21 columns]
```

[ ]:
```
```

1 Data@ANZ Programme Task2

1.1 Predictive Analytics Task

1.1.1 Reindexing

[60]:
```
#print(Anz_dataset['status'])
```

[61]:
```
#Anz_data=Anz_dataset.set_index('status')
```

1.1.2 Creating more features

1.1.3 Feature Annual Salary

[62]:
```
#setting the background for seaborn
sns.set_style("ticks")
#exclude all except pay/salary in txn_description
Salary = Anz_dataset[(Anz_dataset["txn_description"] == "PAY/SALARY")]

#summing amount by customer id to sum up         all the salary during the 3␣
↪months Salary=
Salary.pivot_table(index="customer_id",values="amount",aggfunc=np.sum)
Salary.reset_index(inplace=True)
#creating annual  salary  by  multiplying  the  sum of  amount by  4(the ␣
↪data  is 3 months data and so a year which is 12 months need to be␣
↪multiplied by 4)
Salary["Annual_Salary"] = (round(Salary["amount"]*4,2))
Salary.head()
```

[62]:
```
   index  status  card_present_flag         account  currency        long_lat  \
0     50  posted                NaN   ACC-588564840       AUD  151.27 -33.76
1     61  posted                NaN  ACC-1650504218       AUD  145.01 -37.93
2     64  posted                NaN  ACC-3326339947       AUD  151.18 -33.80
3     68  posted                NaN  ACC-3541460373       AUD  145.00 -37.83
4     70  posted                NaN  ACC-2776252858       AUD  144.95 -37.76

  txn_description merchant_id first_name  balance  ... merchant_suburb  \
0      PAY/SALARY         NaN     Isaiah  8342.11  ...             NaN
1      PAY/SALARY         NaN    Marissa  2040.58  ...             NaN
2      PAY/SALARY         NaN       Eric  3158.51  ...             NaN
3      PAY/SALARY         NaN    Jeffrey  2517.66  ...             NaN
4      PAY/SALARY         NaN    Kristin  2271.79  ...             NaN

  merchant_state                    extraction   amount  \
0            NaN  2018-08-01T11:00:00.000+0000  3903.95
1            NaN  2018-08-01T12:00:00.000+0000  1626.48
2            NaN  2018-08-01T12:00:00.000+0000   983.36
3            NaN  2018-08-01T13:00:00.000+0000  1408.08
4            NaN  2018-08-01T13:00:00.000+0000  1068.04

                     transaction_id    country     customer_id  \
0  9ca281650e5d482d9e53f85e959baa66  Australia  CUS-1462656821
```

```
1  1822eb0e1bbe4c9e95ebbb0fa2cc4323   Australia   CUS-2500783281
2   bd62b1799a454cedbbb56364f7c40cbf   Australia    CUS-326006476
3  0d95c7c932bb48e5b44c2637bdd3efe9   Australia   CUS-1433879684
4 f50ccf1195214d14a0acbfcb5a265193 Australia CUS-4123612273

  merchant_long_lat movement Annual_Salary
0               NaN   credit      15615.80
1               NaN   credit       6505.92
2               NaN   credit       3933.44
3               NaN   credit       5632.32
4               NaN   credit       4272.16

[5 rows x 23 columns]
```

1.1.4 Feature Annual Balance

[63]:
```python
#exclude all except pay/salary in txn_description
Balance = Anz_dataset[(Anz_dataset['txn_description'] == 'PAY/SALARY')]
#summing amount by customer id to sum up all the balance during the 3 months
Balance= Balance.pivot_table(index='customer_id',values='balance',aggfunc=np.sum)
Balance.reset_index(inplace=True)
```

[64]:
```python
#creating annual balance by multiplying the sum of amount by 4(the data is 3 months data and so a year which is 12 months need to be multiplied by 4)
Balance['Annual_balance'] = (round(Balance['balance']*4,2))
Balance.head()
```

[64]:

	customer_id	balance	Annual_balance
0	CUS-1005756958	61342.65	245370.60
1	CUS-1117979751	83700.42	334801.68
2	CUS-1140341822	35050.32	140201.28
3	CUS-1147642491	114575.08	458300.32
4	CUS-1196156254	166920.02	667680.08

1.1.5 Feature Expenses

[65]:
```python
#exclude credit in movement.
Expense = Anz_dataset[Anz_dataset['movement'] == 'debit']
#summing amount by customer id to sum up the expenses during the 3 months
Expense = Expense.pivot_table(index='customer_id',values='amount',aggfunc=np.sum)
#creating annual expenses by multiplying the sum of amount by 4(the data is 3 months data and so a year which is 12 months need to be multiplied by 4)
```

```
Expense["Annual_Expense"] = round(Expense["amount"]*4)
Expense.reset_index(inplace=True)
Expense.head()
```

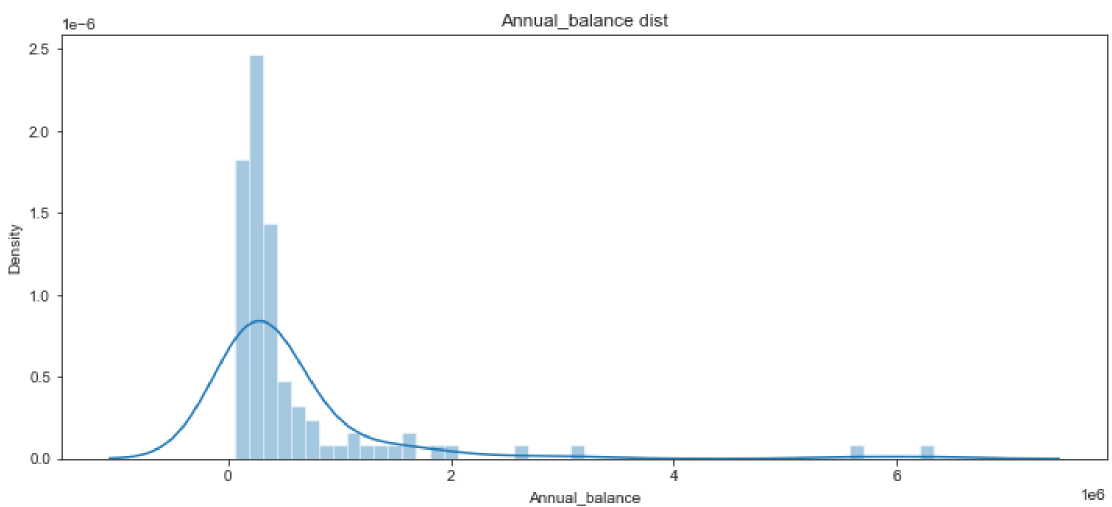
[65]:

	customer_id	amount	Annual_Expense
0	CUS-1005756958	3652.86	14611.0
1	CUS-1117979751	8933.82	35735.0
2	CUS-1140341822	5511.54	22046.0
3	CUS-1147642491	6732.75	26931.0
4	CUS-1196156254	8724.61	34898.0

1.2 Distribution Plots

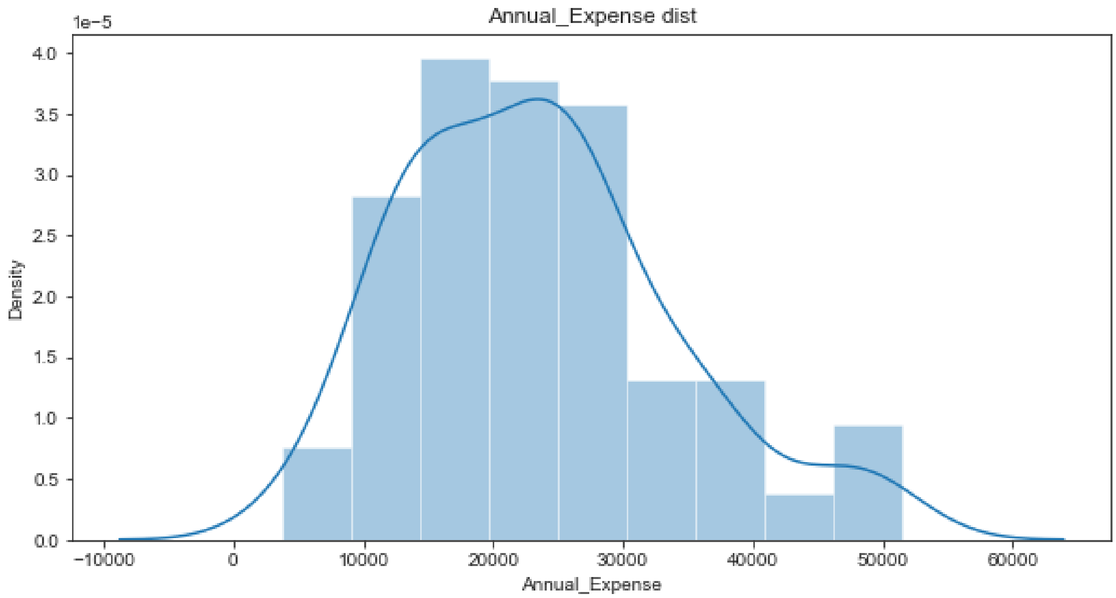
[66]:
```
plt.figure(figsize = (12,5))
sns.distplot(Balance.Annual_balance)
plt.title("Annual_balance dist")
```

[66] : Text(0.5, 1.0, 'Annual_balance dist')



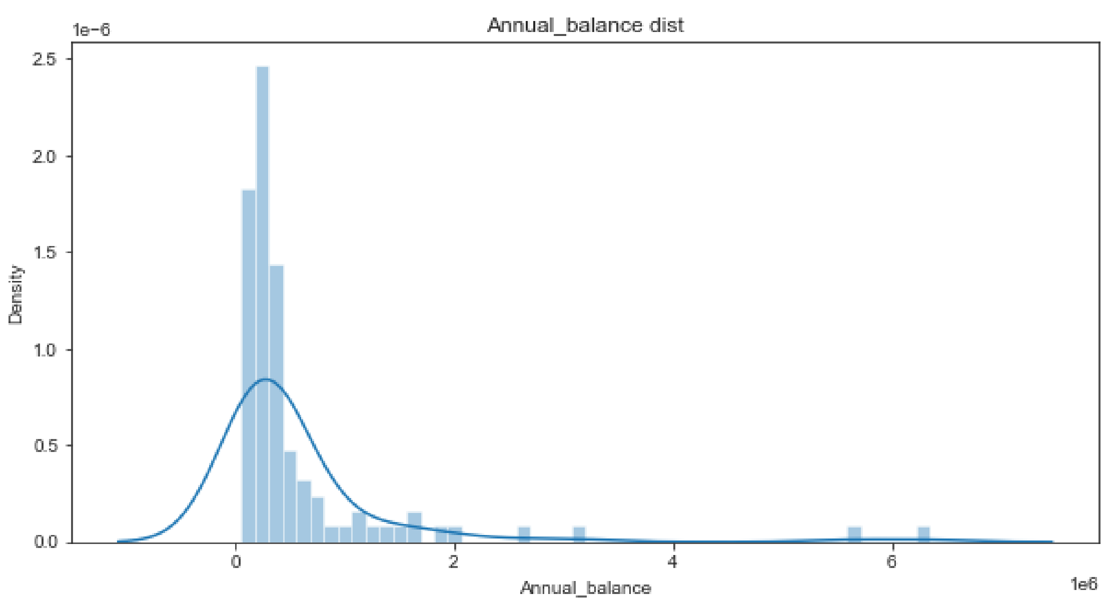
[67]:
```
plt.figure(figsize = (10,5))
sns.distplot(Expense.Annual_Expense)
plt.title("Annual_Expense dist")
```

[67] : Text(0.5, 1.0, 'Annual_Expense dist')



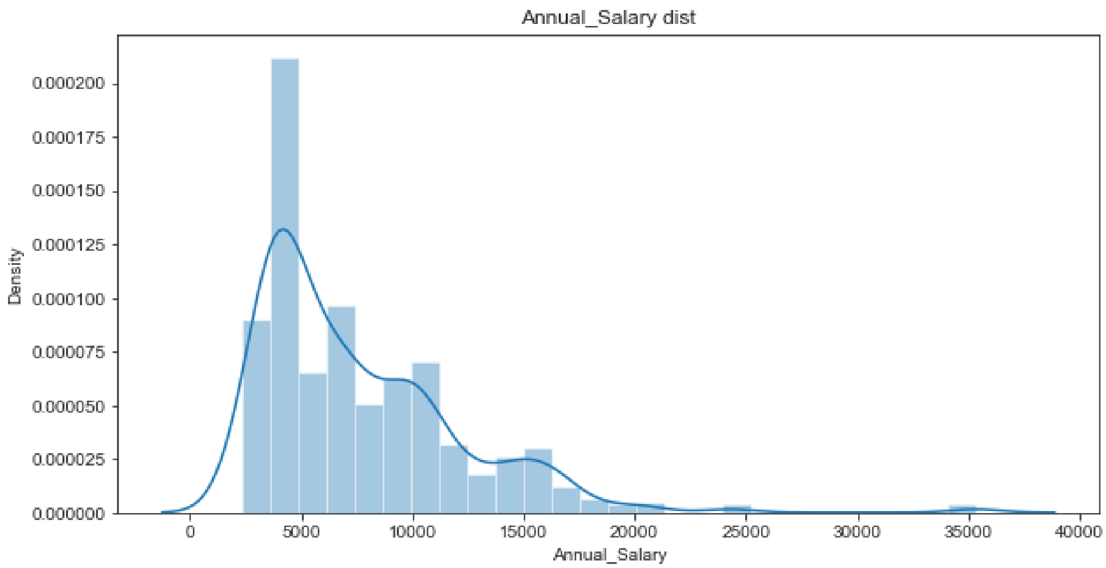
[68]:
```
plt.figure(figsize = (10,5))
sns.distplot(Balance.Annual_balance)
plt.title("Annual_balance  dist")
```

[68] : Text(0.5, 1.0, 'Annual_balance dist')



[69]:
```
plt.figure(figsize = (10,5))
sns.distplot(Salary.Annual_Salary)
plt.title("Annual_Salary dist")
```

[69] : Text(0.5, 1.0, 'Annual_Salary dist')



Merging age and gender with new feaures created

[70] :
```
Salary=pd.
↪merge(Salary,Anz_dataset[["customer_id","age","gender"]],on="customer_id",how="left")
Salary.drop_duplicates(inplace=True)
Salary.head()
```

[70] :
```
     index  status  card_present_flag         account currency       long_lat  \
0       50  posted                NaN   ACC-588564840      AUD  151.27 -33.76
116     61  posted                NaN  ACC-1650504218      AUD  145.01 -37.93
200     64  posted                NaN  ACC-3326339947      AUD  151.18 -33.80
247     68  posted                NaN  ACC-3541460373      AUD  145.00 -37.83
402     70  posted                NaN  ACC-2776252858      AUD  144.95 -37.76

    txn_description merchant_id first_name  balance  ...  \
0       PAY/SALARY         NaN     Isaiah  8342.11  ...
116     PAY/SALARY         NaN    Marissa  2040.58  ...
200     PAY/SALARY         NaN       Eric  3158.51  ...
247     PAY/SALARY         NaN    Jeffrey  2517.66  ...
402     PAY/SALARY         NaN    Kristin  2271.79  ...

                    extraction   amount                   transaction_id  \
```

```
0     2018-08-01T11:00:00.000+0000   3903.95  9ca281650e5d482d9e53f85e959baa66
116   2018-08-01T12:00:00.000+0000   1626.48  1822eb0e1bbe4c9e95ebbb0fa2cc4323
200   2018-08-01T12:00:00.000+0000    983.36  bd62b1799a454cedbbb56364f7c40cbf
247   2018-08-01T13:00:00.000+0000   1408.08  0d95c7c932bb48e5b44c2637bdd3efe9
402   2018-08-01T13:00:00.000+0000   1068.04  f50ccf1195214d14a0acbfcb5a265193

       country    customer_id merchant_long_lat movement  Annual_Salary  \
0    Australia  CUS-1462656821               NaN   credit       15615.80
116  Australia  CUS-2500783281               NaN   credit        6505.92
200  Australia   CUS-326006476               NaN   credit        3933.44
247  Australia  CUS-1433879684               NaN   credit        5632.32
402  Australia  CUS-4123612273               NaN   credit        4272.16

     age_y gender_y
0       23        M
116     23        F
200     22        M
247     24        M
402     43        F

[5 rows x 25 columns]
```

1.2.1 Merging balance data with salary data

[71]:
```python
Salary = pd.merge(Salary,Expense,on="customer_id",how="left")
Salary.drop_duplicates(inplace=True)
Salary.head()
```

[71]:
```
   index  status  card_present_flag         account currency        long_lat  \
0     50  posted                NaN   ACC-588564840      AUD  151.27 -33.76
1     61  posted                NaN  ACC-1650504218      AUD  145.01 -37.93
2     64  posted                NaN  ACC-3326339947      AUD  151.18 -33.80
3     68  posted                NaN  ACC-3541460373      AUD  145.00 -37.83
4     70  posted                NaN  ACC-2776252858      AUD  144.95 -37.76

  txn_description merchant_id first_name  balance  ...  \
0      PAY/SALARY         NaN     Isaiah  8342.11  ...
1      PAY/SALARY         NaN    Marissa  2040.58  ...
2      PAY/SALARY         NaN       Eric  3158.51  ...
3      PAY/SALARY         NaN    Jeffrey  2517.66  ...
4      PAY/SALARY         NaN    Kristin  2271.79  ...

                     transaction_id    country     customer_id  \
0 9ca281650e5d482d9e53f85e959baa66 Australia CUS-1462656821
1  1822eb0e1bbe4c9e95ebbb0fa2cc4323  Australia  CUS-2500783281
2  bd62b1799a454cedbbb56364f7c40cbf  Australia   CUS-326006476
3  0d95c7c932bb48e5b44c2637bdd3efe9  Australia  CUS-1433879684
```

```
4  f50ccf1195214d14a0acbfcb5a265193  Australia   CUS-4123612273

  merchant_long_lat  movement  Annual_Salary   age_y gender_y amount_y   \
0               NaN    credit       15615.80      23        M  7183.55
1               NaN    credit        6505.92      23        F  3637.50
2               NaN    credit        3933.44      22        M  1920.93
3               NaN    credit        5632.32      24        M  5706.80
4               NaN    credit        4272.16      43        F  7359.56

  Annual_Expense
0        28734.0
1        14550.0
2         7684.0
3        22827.0
4        29438.0

[5 rows x 27 columns]
```

1.2.2 Merging Expense to data

[72]:
```python
Salary=pd.merge(Salary,Expense,on="customer_id",  how="left")
Salary.drop_duplicates(inplace=True)
Salary.head()
```

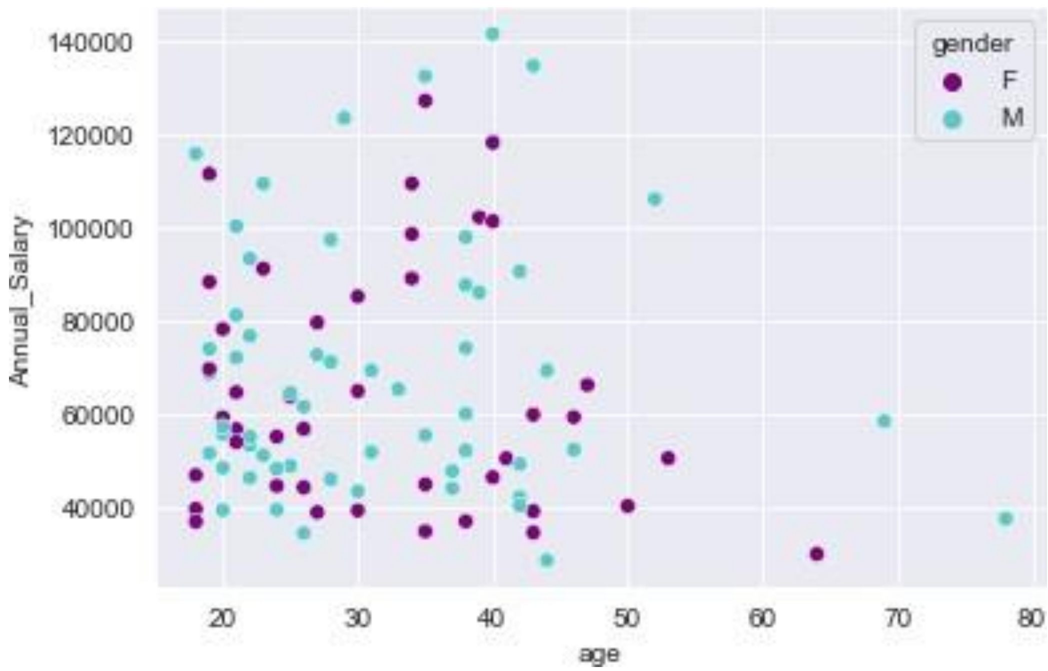
[72]:
```
   index  status  card_present_flag         account  currency        long_lat   \
0     50  posted                NaN   ACC-588564840       AUD  151.27 -33.76
1     61  posted                NaN  ACC-1650504218       AUD  145.01 -37.93
2     64  posted                NaN  ACC-3326339947       AUD  151.18 -33.80
3     68  posted                NaN  ACC-3541460373       AUD  145.00 -37.83
4     70  posted                NaN  ACC-2776252858       AUD  144.95 -37.76

  txn_description  merchant_id  first_name  balance  ...     customer_id   \
0      PAY/SALARY          NaN      Isaiah  8342.11  ...  CUS-1462656821
1      PAY/SALARY          NaN     Marissa  2040.58  ...  CUS-2500783281
2      PAY/SALARY          NaN        Eric  3158.51  ...   CUS-326006476
3      PAY/SALARY          NaN     Jeffrey  2517.66  ...  CUS-1433879684
4      PAY/SALARY          NaN     Kristin  2271.79  ...  CUS-4123612273

  merchant_long_lat  movement Annual_Salary age_y gender_y  amount_y  \
0               NaN    credit      15615.80    23        M   7183.55
1               NaN    credit       6505.92    23        F   3637.50
2               NaN    credit       3933.44    22        M   1920.93
3               NaN    credit       5632.32    24        M   5706.80
4               NaN    credit       4272.16    43        F   7359.56

  Annual_Expense_x   amount Annual_Expense_y
0          28734.0  7183.55          28734.0
```

1	4	14550.0	29437.50	7359.56	14550.0 29438.0
2		7684.0	1920.93		7684.0
3		22827.0	5706.80		22827.0

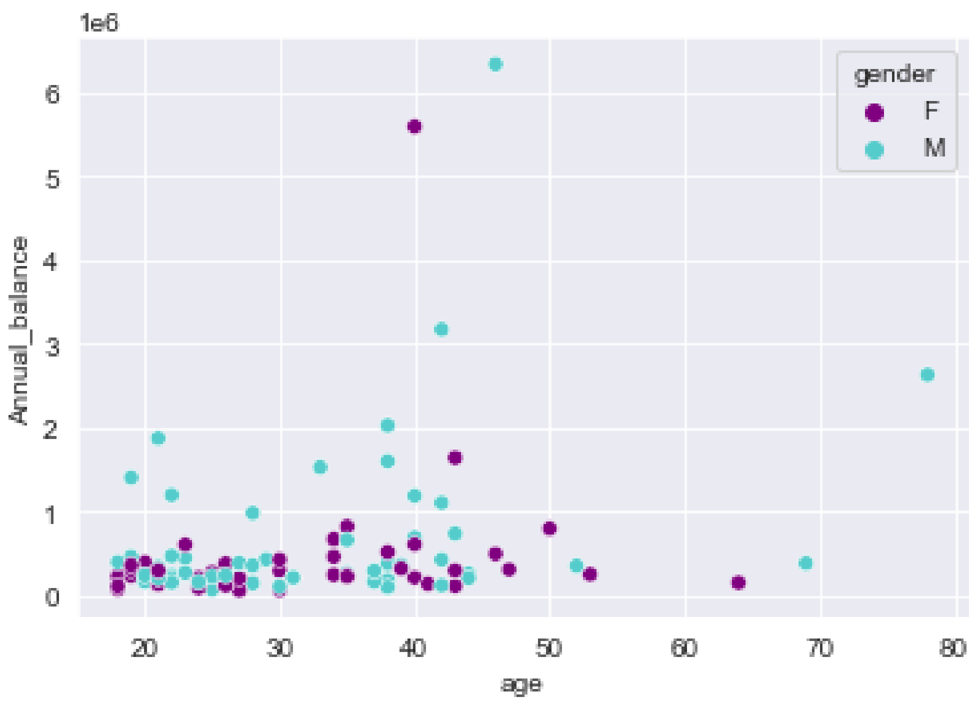
[5 rows x 29 columns]

Scatter plot of annual salary with other features

```
[73]: sns.scatterplot(x= "age", y= "Annual_Salary",  hue="gender",
                      Anz_dataset =Salary,
                      palette=["purple", "#55CCCC"]);
```



[74]:
```
sns.scatterplot(x= 'age', y= 'Annual_balance', hue='gender',
                Anz_dataset=Salary,
                palette=['purple', '#55CCCC']);
```



Dropping irrelevant features

[75]:
```
Salary.drop(columns =["customer_id","amount_x","amount_y","balance"], inplace=
↪True)
```

Checking Correlation

[76]:
```
Salary.corr()
```

[76]:

	Annual_Salary	age	Annual_balance	Annual_Expense
Annual_Salary	1.000000	-0.036504	0.217715	0.373476
age	-0.036504	1.000000	0.289224	-0.086174
Annual_balance	0.217715	0.289224	1.000000	0.004943
Annual_Expense	0.373476	-0.086174	0.004943	1.000000

The data looks good with low correlation.

[77]:
```
Salary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 0 to 99
Data columns (total 5 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  ------
 0   Annual_Salary   100 non-null    float64
 1   age             100 non-null    int64
 2   gender          100 non-null    object
 3   Annual_balance  100 non-null    float64
 4   Annual_Expense  100 non-null    float64
dtypes: float64(3), int64(1), object(1)
memory usage: 4.7+ KB
```

One Hot Encoding of gender

[78]:
```
Salary = pd.get_dummies(Salary,columns=["gender"])
```

[79]:
```
Salary.head()
```

[79]:

	Annual_Salary	age	Annual_balance	Annual_Expense	gender_F	gender_M
0	50464.44	53	245370.60	14611.0	1	0
1	100202.20	21	334801.68	35735.0	0	1

```
2       45996.24    28        140201.28         22046.0          0          1
3       88992.28    34        458300.32         26931.0          1          0
4      109304.44    34        667680.08         34898.0          1          0
```

[80]:
```
Salary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype
---  -------          --------------  ------
 0   Annual_Salary    100 non-null    float64
 1   age              100 non-null    int64
 2   Annual_balance   100 non-null    float64
 3   Annual_Expense   100 non-null    float64
 4   gender_F         100 non-null    uint8
 5   gender_M         100 non-null    uint8
dtypes: float64(3), int64(1), uint8(2)
memory usage: 4.1 KB
```

Linear regression Model

[81]:
```
x = Salary.drop("Annual_Salary", axis="columns")
y = Salary["Annual_Salary"]
```

[82]:
```
#initialize the linear regression model
reg = linear_model.LinearRegression()
```

[83]:
```
#split the data into 70% training and 33% testing data
x_train, x_test, y_train, y_test = train_test_split(x,y, test_size=0.
↪3,random_state=7)
```

[84]:
```
#train the model
reg.fit(x_train, y_train)
```

[84]:
```
LinearRegression()
```

[85]:
```
#print the coefficients/weights for each feature/column of our model
print(reg.coef_)
```

```
[-2.19854918e+02  2.98116645e-03  8.77244212e-01 -3.24159489e+03
  3.24159489e+03]
```

[86]:
```
y_pred = reg.predict(x_test)
print(y_pred)
```

```
[66169.19169742  73238.40211721  67870.27501126  82291.76186935
 58072.28717096  57881.30249663  69273.84842515  57906.52412454
```

```
 67626.99063658 75146.47780116 77691.48793785 64701.99935058
 65233.22702201 71287.20044804 62880.56730571 62093.20570672
 56473.86099523 61340.1809038  75227.34041434 68533.01201909
 71981.9013954  69816.96105331 78150.49074061 54270.48300664
 61215.47635446 75355.5409963  54143.23057579 73745.14836422
 76595.75997825  64676.66144645]
```

[87]:
```python
#print the actual values
print(y_test)
```

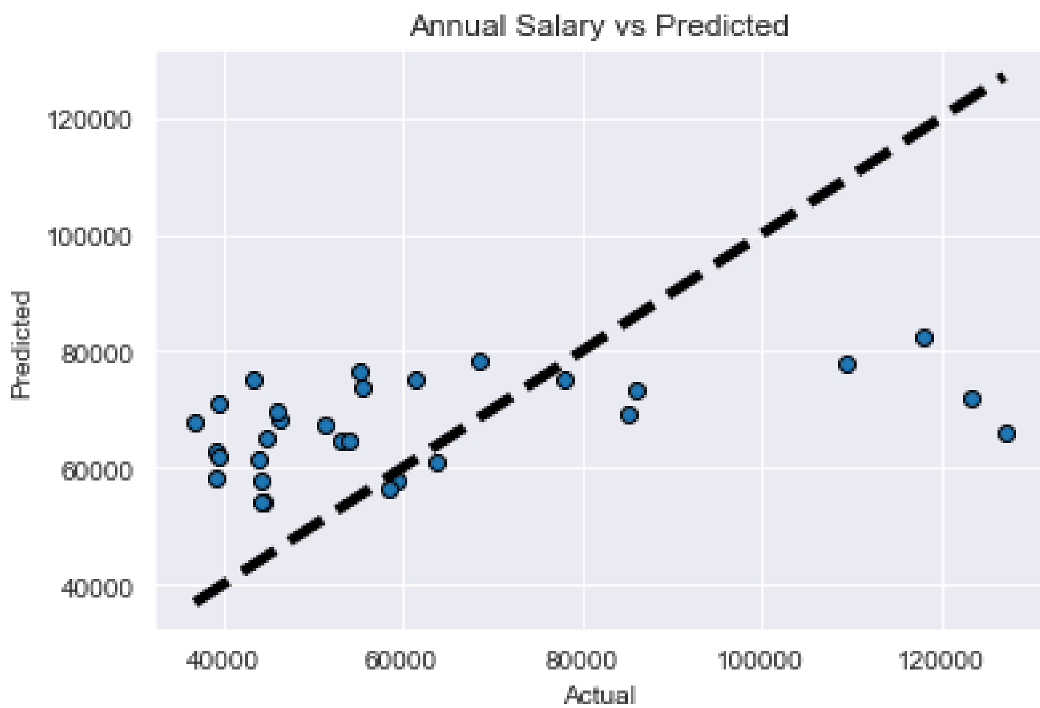
```
37    127048.48
26     85991.92
78     36904.32
91    118049.12
49     39128.64
15     59290.80
93     85109.44
71     44235.36
86     51508.60
22     61538.96
13    109310.60
40     53249.52
52     44873.40
12     39426.24
88     39287.20
45     39379.92
11     58414.72
66     44004.00
20     78147.44
18     46388.68
50    123348.40
2      45996.24
17     68513.76
85     44550.72
5      63906.08
97     43406.88
51     44194.08
30     55408.08
74     55111.68
96     53927.64
Name:  Annual_Salary, dtype: float64
```

[88]:
```python
reg.score(x_train,y_train)
```

[88]: 0.1741029921866517

[89]:
```python
fig, ax = plt.subplots()
ax.scatter(y_test, y_pred, edgecolors=(0, 0, 0))
```

```
ax.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=4)
ax.set_xlabel('Actual')
ax.set_ylabel('Predicted')
ax.set_title("Annual Salary vs Predicted")
plt.show()
```



The linear regression model accuracy was 17.4%

DECISION TREE

[90]:
```
x = Salary.drop('Annual_Salary', axis='columns')
y = Salary['Annual_Salary']
```

[91]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,
test_size=0.3,random_state=7)
```

[92]:
```
dt = DecisionTreeRegressor(max_depth=4,min_samples_leaf=0.1,random_state=7)
```

[93]:
```
dt.fit(x_train, y_train)
```

[93]: DecisionTreeRegressor(max_depth=4, min_samples_leaf=0.1, random_state=7)

[94]:
```
y_pred = dt.predict(x_test)
print(y_pred)
```

```
[ 76408.79428571  96712.74285714  52270.156       96712.74285714
  52270.156       96712.74285714  76408.79428571  44048.15076923
 64360.90153846  76408.79428571  76408.79428571  44048.15076923
 70733.33333333  52270.156       52270.156       44048.15076923
 64360.90153846  44048.15076923 106748.31        70733.33333333
 76408.79428571  52270.156       76408.79428571  44048.15076923
 70733.33333333  52270.156       64360.90153846  70733.33333333
 70733.33333333  70733.33333333]
```

[95]:
```
print(y_test)
```

```
37    127048.48
26     85991.92
78     36904.32
91    118049.12
49     39128.64
15     59290.80
93     85109.44
71     44235.36
86     51508.60
22     61538.96
13    109310.60
40     53249.52
52     44873.40
12     39426.24
88     39287.20
45     39379.92
11     58414.72
66     44004.00
20     78147.44
18     46388.68
50    123348.40
2      45996.24
17     68513.76
85     44550.72
5      63906.08
97     43406.88
51     44194.08
30     55408.08
74     55111.68
96     53927.64
Name: Annual_Salary, dtype: float64
```

[96]:
```
dt.score(x_train,y_train)
```

[96]: 0.5600863941994056

[97]:
```
mse_dt = MSE(y_test, y_pred)
```

[98]:
```python
rmse_dt = mse_dt**(1/2)
```

[99]:
```python
print(rmse_dt)
```

```
20527.660154490288
```

Decision Tree Regression accuracy was 56%.

Conclusion

The decision was the preferred algorithm for now because it more efficient than linear regression. Linear regression accuracy was 17 percent while that of decision tree was 56 percent. We may use random forest model or XGBoost Model subsequently to see their performance in predicting customers' incomes bracket which is an important feature in this case,