Problem Statement 2 (Predict expected revenue from a customer)

The CRM team of a retail store wants to:

1. Understand which factors influence the revenue generated by the customers

2. Predict the revenue "Amount" for a given customer is likely to generate

```
# Importing libraries

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning:
  import pandas.util.testing as tm
```

```
# Reading the dataset

data=pd.read_csv("RetailCustomerRevenue.csv")
```

```
# Obtaining the column names

data.columns
```

```
Index(['PersonID', 'Amount', 'FamilySize', 'Distance', 'Duration',
       'DirectVisits', 'OnlineVisits', 'Quantity', 'NumberofFrequentItems',
       'TransactionMode', 'Area', 'Occupation'],
      dtype='object')
```

```
# Reading the first few rows of data

data.head()
```

	PersonID	Amount	FamilySize	Distance	Duration	DirectVisits	OnlineVisits	Quanti
0	C1104	3125	2	6	261	11	9	3
1	C1111	5298	2	5	323	9	9	2
2	C1117	4375	2	6	355	11	11	13
3	C1128	9700	5	7	418	51	41	29
4	C1132	3625	2	7	290	9	9	12

```
# Obtaining the dimensions

data.shape
```

```
(2938, 12)
```

```
# Obtaining the data types

data.dtypes
```

```
PersonID                 object
Amount                    int64
FamilySize                int64
Distance                  int64
Duration                  int64
DirectVisits              int64
OnlineVisits              int64
Quantity                  int64
NumberofFrequentItems     int64
TransactionMode           int64
Area                     object
Occupation                int64
dtype: object
```

```
# Summary Statistics

data.describe(include='all')
```

	PersonID	Amount	FamilySize	Distance	Duration	DirectVisits	Onl:
count	2938	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	29
unique	2938	NaN	NaN	NaN	NaN	NaN	
top	C3960	NaN	NaN	NaN	NaN	NaN	
freq	1	NaN	NaN	NaN	NaN	NaN	

```
# Obtaining the variables having unique values less than 10

ins=[]
for a in data.columns:
  if(data[a].nunique()<10):
    ins.append(a)
    print(a,data[a].nunique())
    print(data[a].unique())
```

```
TransactionMode 2
[2 1]
Area 2
['Area1' 'Area2']
Occupation 3
[2 1 3]
```

```
ins
```

```
['TransactionMode', 'Area', 'Occupation']
```

```
# Checking for missing values

data.isna().sum()
```

```
PersonID                 0
Amount                   0
FamilySize               0
Distance                 0
Duration                 0
DirectVisits             0
OnlineVisits             0
Quantity                 0
NumberofFrequentItems    0
TransactionMode          0
Area                     0
Occupation               0
dtype: int64
```

```
# Converting required data types into categorical

for col in ['TransactionMode', 'Area', 'Occupation','PersonID']:
  data[col]=data[col].astype('category')
```

```
# Checking to see if type has been converted

data.dtypes
```

```
PersonID                  category
Amount                       int64
FamilySize                   int64
Distance                     int64
Duration                     int64
DirectVisits                 int64
OnlineVisits                 int64
Quantity                     int64
NumberofFrequentItems        int64
TransactionMode           category
Area                      category
Occupation                category
dtype: object
```

```
# Dropping duplicate values

data=data.drop_duplicates(keep='first')
```

```
# Dimensions of dataset

data.shape
```

```
(2938, 12)
```

There are no duplicate values as the dimensions have not changed.

```
# Obtaining the correlation between each of the attributes

data.corr()
```

	Amount	FamilySize	Distance	Duration	DirectVisits	OnlineV
Distance	-0.000649	0.436628	1.000000	-0.057219	-0.004709	-0.0

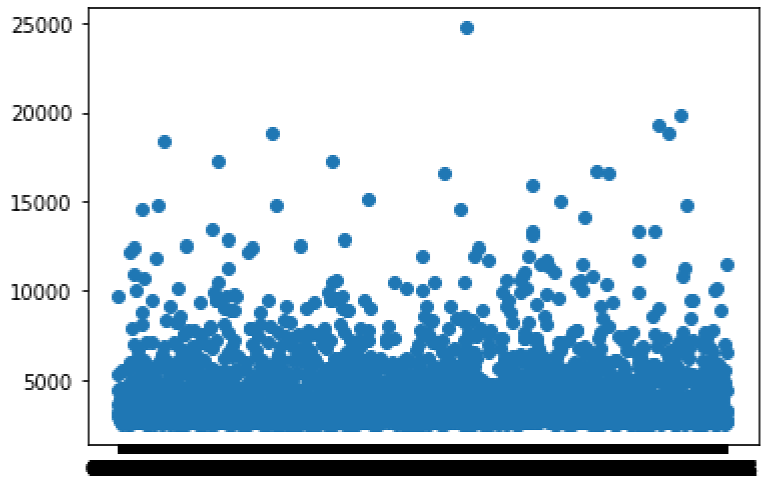
```
# taking dependent variable "Amount" into y

y=np.array(data['Amount'])
```

Obtaining Scatter plots between each attribute with y

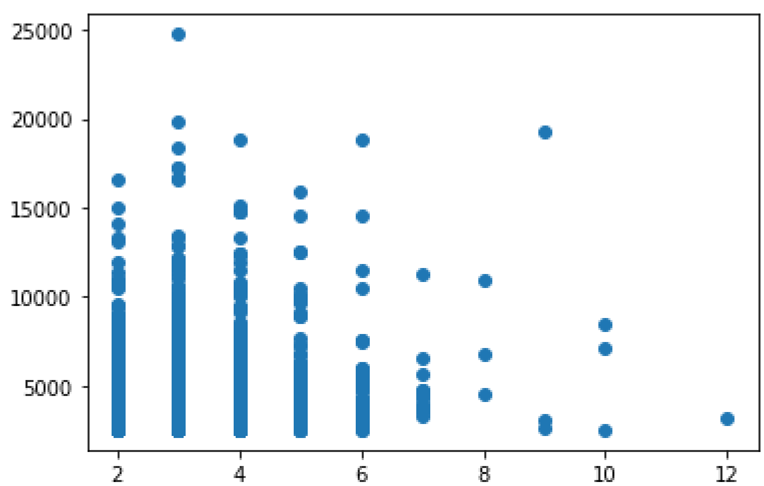
```
plt.scatter(data['PersonID'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe0219af3c8>
```



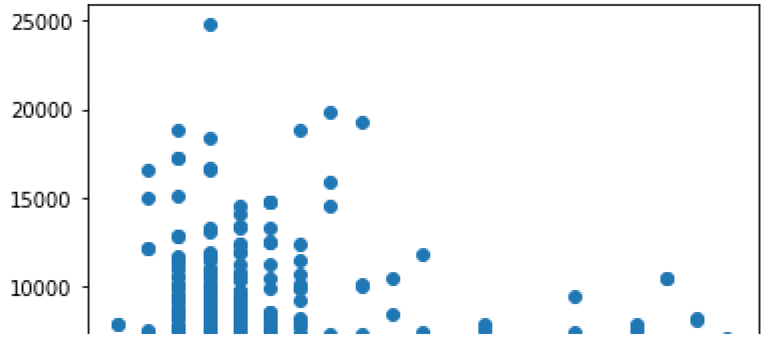

```
plt.scatter(data['FamilySize'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe01fa3ecf8>
```



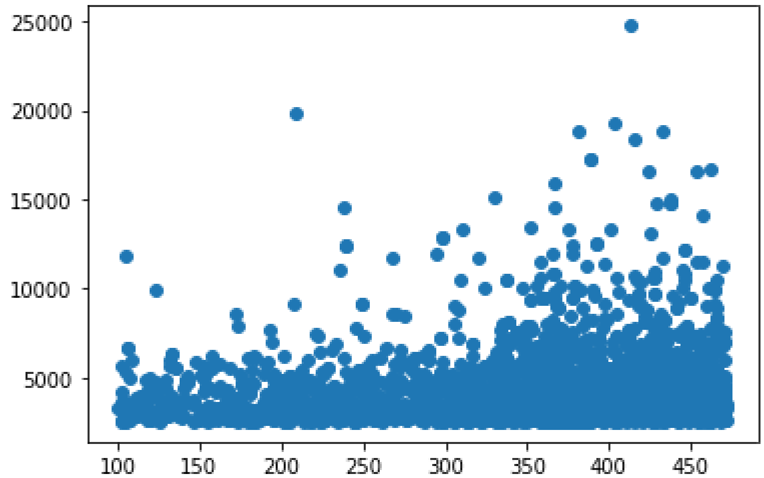

```
plt.scatter(data['Distance'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe02006c748>
```



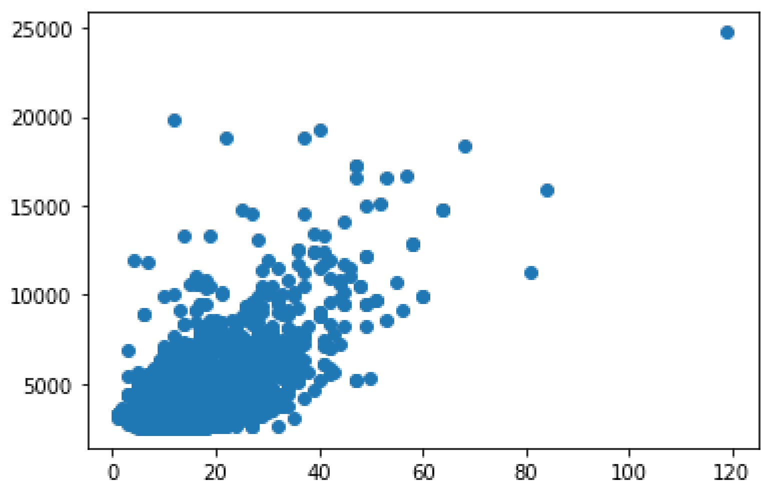
```
plt.scatter(data['Duration'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe01fad8cc0>
```



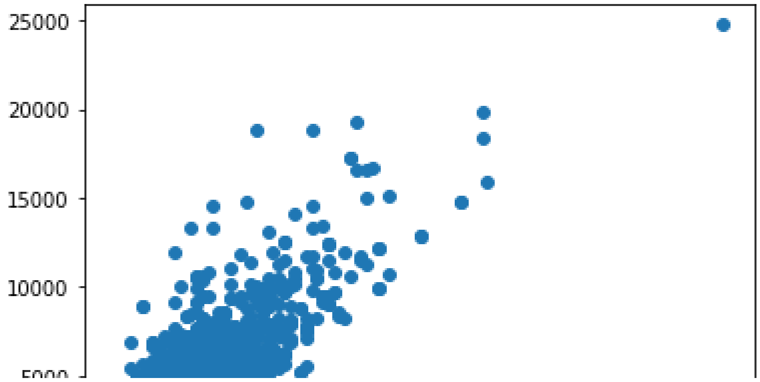
```
plt.scatter(data['DirectVisits'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe01fa10978>
```



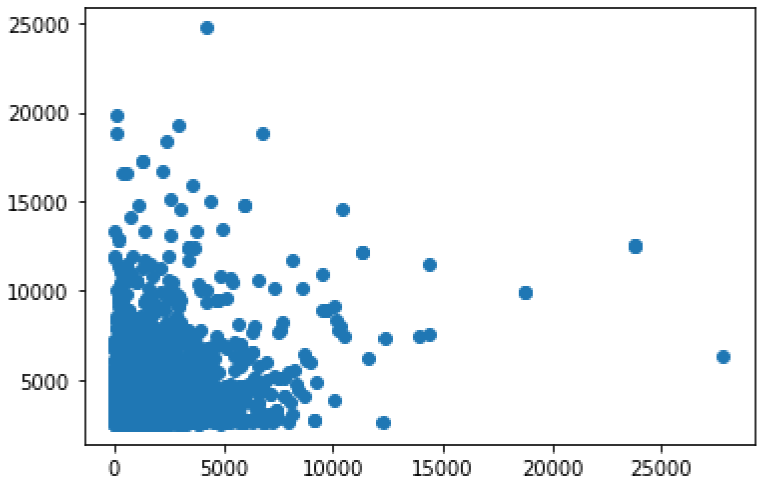
```
plt.scatter(data['OnlineVisits'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe020388a90>
```



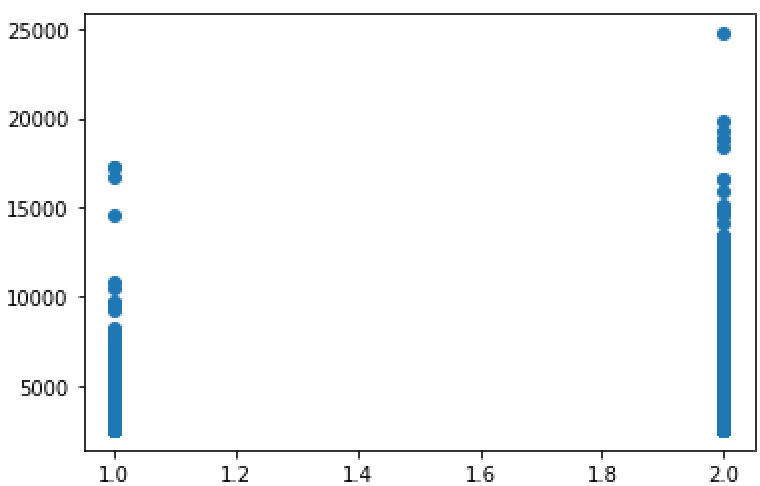

```
plt.scatter(data['Quantity'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe01f415588>
```



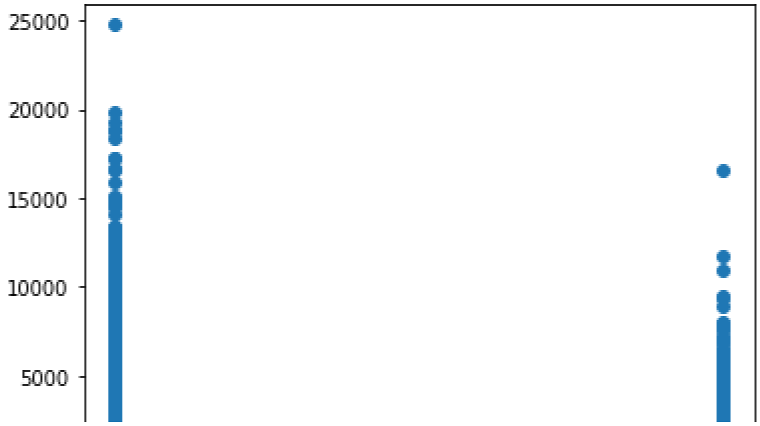

```
plt.scatter(data['TransactionMode'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe020210fd0>
```



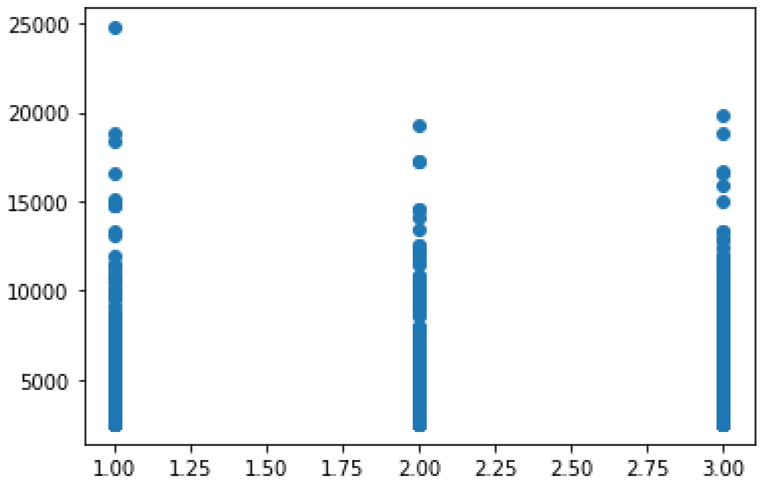

```
plt.scatter(data['Area'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe020333c18>
```




```
plt.scatter(data['Occupation'],y)
```

```
<matplotlib.collections.PathCollection at 0x7fe01fa385c0>
```



According to the scatter plots, "TransactionMode", " Area" and "Occupation" are categorical.

```
# Splitting into train and test data

train_data,test_data=train_test_split(data,test_size=0.2,random_state=123)
```

```
# Printing the dimensions of the train and test data

print(train_data.shape)
print(test_data.shape)
```

```
(2350, 12)
(588, 12)
```

```
# Creating dummy variables

train_dum=pd.get_dummies(columns=['TransactionMode','Area', 'Occupation'],data=train_data)
test_dum=pd.get_dummies(columns=['TransactionMode','Area', 'Occupation'],data=test_data)
```

```
# Printing the columns of the data after creating the dummy variables

print(train_dum.columns)
print(test_dum.columns)
```

```
Index(['PersonID', 'Amount', 'FamilySize', 'Distance', 'Duration',
       'DirectVisits', 'OnlineVisits', 'Quantity', 'NumberofFrequentItems',
       'TransactionMode_1', 'TransactionMode_2', 'Area_Area1', 'Area_Area2',
       'Occupation_1', 'Occupation_2', 'Occupation_3'],
      dtype='object')
Index(['PersonID', 'Amount', 'FamilySize', 'Distance', 'Duration',
       'DirectVisits', 'OnlineVisits', 'Quantity', 'NumberofFrequentItems',
       'TransactionMode_1', 'TransactionMode_2', 'Area_Area1', 'Area_Area2',
       'Occupation_1', 'Occupation_2', 'Occupation_3'],
      dtype='object')
```

```
train_data1=train_dum
test_data1=test_dum
```

```
# Dropping "PersonID" attribute

train_data1=train_dum.drop('PersonID',axis=1)
test_data1=test_dum.drop('PersonID',axis=1)
```

```
print(train_data1.columns)
print(test_data1.columns)
```

```
Index(['Amount', 'FamilySize', 'Distance', 'Duration', 'DirectVisits',
       'OnlineVisits', 'Quantity', 'NumberofFrequentItems',
       'TransactionMode_1', 'TransactionMode_2', 'Area_Area1', 'Area_Area2',
       'Occupation_1', 'Occupation_2', 'Occupation_3'],
      dtype='object')
Index(['Amount', 'FamilySize', 'Distance', 'Duration', 'DirectVisits',
       'OnlineVisits', 'Quantity', 'NumberofFrequentItems',
       'TransactionMode_1', 'TransactionMode_2', 'Area_Area1', 'Area_Area2',
       'Occupation_1', 'Occupation_2', 'Occupation_3'],
      dtype='object')
```

```
# Placing the independent variables into ind_atr

ind_atr=list(set(train_data1.columns)-set(['Amount']))
```

```
ind_atr
```

```
['Occupation_3',
 'Duration',
 'OnlineVisits',
 'Area_Area1',
 'FamilySize',
 'Area_Area2',
 'DirectVisits',
 'Occupation_1',
 'TransactionMode_1',
 'Quantity',
 'Occupation_2',
```

```
# Fitting LinearRegression model

linreg=LinearRegression()
res_sklearn=linreg.fit(train_data1[ind_atr],train_data1['Amount'])


# Predicting train and test values from the model

pres_train_skleran=res_sklearn.predict(train_data1[ind_atr])
pres_test_skleran=res_sklearn.predict(test_data1[ind_atr])


# Printing the coefficients

res_sklearn.coef_
```

```
array([-1.37942112e+01, -5.25526037e-01,  2.32211345e+02, -1.51619314e+02,
        4.71029449e+01,  1.51619314e+02,  2.62546923e+02, -1.82477056e+00,
       -1.32104193e+02,  3.84743602e-02,  1.56189818e+01, -9.47350161e+00,
       -2.91464118e+02,  1.32104193e+02])
```

```
# Finding mean absolute error for train data

mean_absolute_error(np.array(train_data1['Amount']),pres_train_skleran)
```

```
793.2396053830885
```

```
# Finding mean absolute error for test data

mean_absolute_error(np.array(test_data1['Amount']),pres_test_skleran)
```

```
778.8292727155123
```

Amount depends on 'FamilySize', 'Distance', 'Duration', 'DirectVisits', 'OnlineVisits', 'Quantity', 'NumberofFrequentItems', 'TransactionMode', 'Area', 'Occupation'

Stats model

```
ind_atr1=list(set(data.columns)-set(['Amount','PersonID']))
```

```
ind_atr1
```

```
['Occupation',
 'Duration',
 'OnlineVisits',
 'FamilySize',
 'Area',
 'DirectVisits',
 'TransactionMode',
 'Quantity',
 'Distance',
 'NumberofFrequentItems']
```

```
#Creating the formula

x='+'.join(ind_atr1)
x
```

```
'Occupation+Duration+OnlineVisits+FamilySize+Area+DirectVisits+TransactionMode+Quantit
y+Distance+NumberofFrequentItems'
```

```
formula="~".join(('Amount',x))
print(formula)
```

```
Amount~Occupation+Duration+OnlineVisits+FamilySize+Area+DirectVisits+TransactionMode+Qu
```

```
from statsmodels.formula.api import ols
```

```
#Model fit and summary

lm_mod=ols(formula=formula,data=data)
result=lm_mod.fit()
```

```
print(result.summary2())
```

```
                    Results: Ordinary least squares
=============================================================================
Model:                 OLS                  Adj. R-squared:       0.716
Dependent Variable:    Amount               AIC:                  49518.8936
Date:                  2020-07-15 11:09     BIC:                  49590.7194
No. Observations:      2938                 Log-Likelihood:       -24747.
Df Model:              11                   F-statistic:          673.3
Df Residuals:          2926                 Prob (F-statistic):   0.00
R-squared:             0.717                Scale:                1.2179e+06
-----------------------------------------------------------------------------
                        Coef.   Std.Err.    t     P>|t|    [0.025    0.975]
-----------------------------------------------------------------------------
Intercept              621.3437 120.6836   5.1485 0.0000  384.7103  857.9771
Occupation[T.2]          4.6484  50.9386   0.0913 0.9273  -95.2308  104.5276
Occupation[T.3]        -13.7002  49.8891  -0.2746 0.7836 -111.5215   84.1212
Area[T.Area2]          267.0128  66.3723   4.0230 0.0001  136.8716  397.1540
TransactionMode[T.2]   299.4014  55.5814   5.3867 0.0000  190.4188  408.3841
Duration                -0.5462   0.2379  -2.2961 0.0217   -1.0126   -0.0798
OnlineVisits           232.7890   8.3187  27.9837 0.0000  216.4778  249.1001
Distance                -4.1675   7.6506  -0.5447 0.5860  -19.1686   10.8335
-----------------------------------------------------------------------------
* The condition number is large (2e+04). This might indicate
strong multicollinearity or other numerical problems.
```

OBSERVATION

The R-Squared error is 0.717. If R-squared value r > 0.7 this value is generally considered strong effect size.