Credit Card Task

By Gerard Utoware

Table of Contents

Overview of the tool used

The main advantage of python in comparison to other data analytic tools such as excel, would be Python's ability to handle large volumes of data without hindering productivity. Running scripts using the libraries on allows for more automation and ease of obtaining analysed data. With this task, the data was imported from python which ensures that data will not be lost or tampered with while performing data analysis. After defining the columns from the imported data, it was less tedious to run scripts than to perform individual analysis on excel.

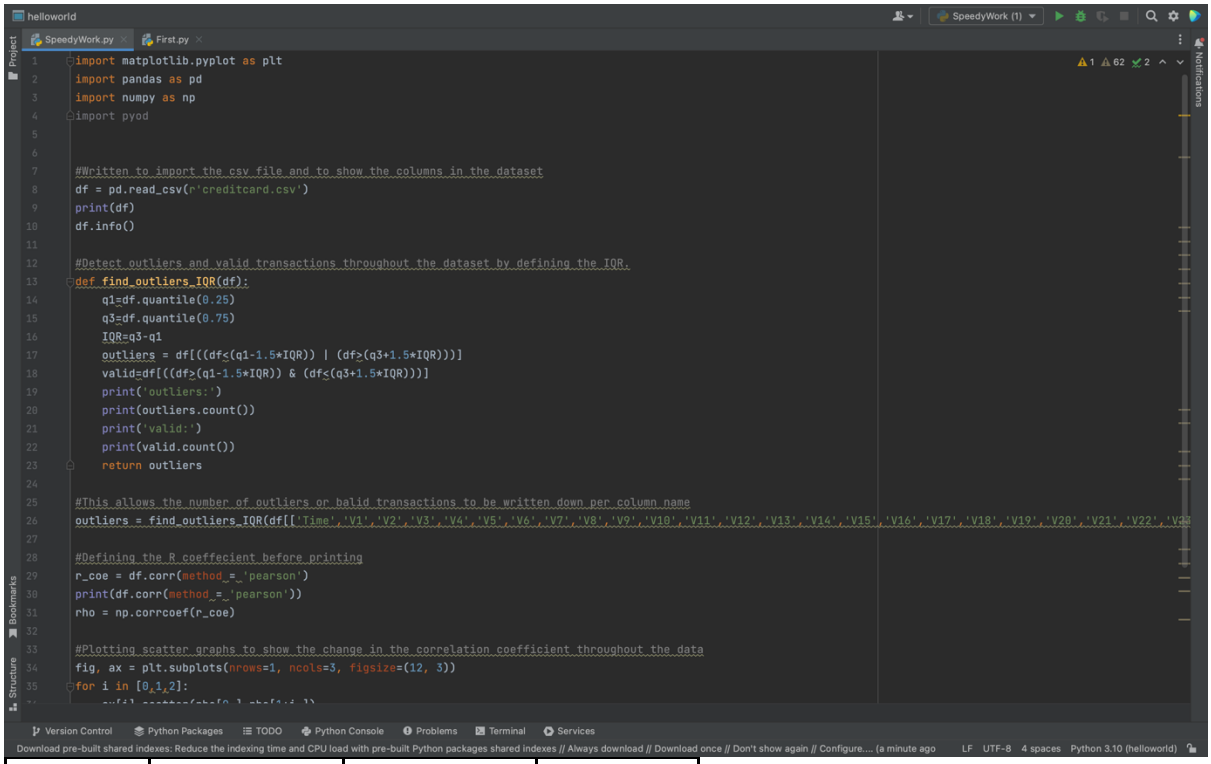
Overview of the libraries used

Numpy, pandas and matplotlib were used in the script. All three python libraries allow different functions to be performed. Pandas allows data analysis scripts to be conducted on python. It utilises two other libraries, being numpy and matplotlib. Matplotlib allows for data visualisation through a variety of charts such as bar and pie charts. Numpy permits mathematical operations in python. The combination of these three libraries is what allowed the following data and charts to be formed and analysed. In addition to these, seaborn was installed in order for a heatmap to be printed and shown.

Overview of the dataset

Overall, the data contains fraudulent data that's been flagged up by the script using python. V8 and V28 are the two columns with the most outliers in the data with 79 and 75 fraudulent transactions respectively. V6 and V11 both only had 1 outlier in their columns. With that being said, the r coefficient is overall stays around 0 which means there is no strong positive or negative correlation in the dataset.

Q1&2



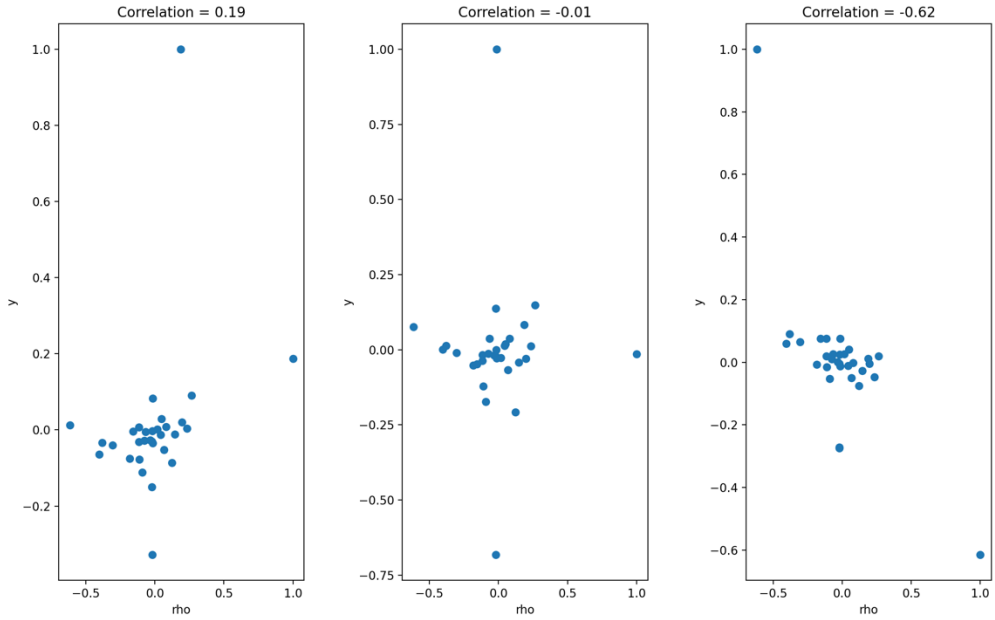
```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import pyod


#Written to import the csv file and to show the columns in the dataset
df = pd.read_csv(r'creditcard.csv')
print(df)
df.info()

#Detect outliers and valid transactions throughout the dataset by defining the IQR.
def find_outliers_IQR(df):
    q1=df.quantile(0.25)
    q3=df.quantile(0.75)
    IQR=q3-q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    valid=df[((df>(q1-1.5*IQR)) & (df<(q3+1.5*IQR)))]
    print('outliers:')
    print(outliers.count())
    print('valid:')
    print(valid.count())
    return outliers

#This allows the number of outliers or balid transactions to be written down per column name
outliers = find_outliers_IQR(df[['Time','V1','V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V14','V15','V16','V17','V18','V19','V20','V21','V22','V23

#Defining the R coeffecient before printing
r_coe = df.corr(method='pearson')
print(df.corr(method='pearson'))
rho = np.corrcoef(r_coe)

#Plotting scatter graphs to show the change in the correlation coefficient throughout the data
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(12, 3))
for i in [0,1,2]:
```

	Fraudulent Transactions (Outliers)	Valid Transactions	Fractional Ratio
Time	0	284807	0
V1	7062	277745	3/121
V2	13526	271281	18/379
V3	3363	281444	3/254
V4	11148	273659	31/792
V5	12295	261842	27/602
V6	22965	275512	1/13
V7	8948	261842	19/575
V8	24134	275859	79/982
V9	8283	260673	17/552
V10	9496	276524	25/753
V11	780	284027	1/365
V12	15348	269459	9/167
V13	3368	281913	10/847
V14	14149	270658	31/624

V15	2894	281913	5/492
V16	8184	276623	5/174
V17	7420	277387	13/499
V18	7533	277274	26/983
V19	10205	274602	11/307
V20	27770	257037	43/441
V21	14497	270310	48/943
V22	1317	283490	4/865
V23	18541	266266	36/553
V24	4774	280033	3/179
V25	5367	279440	15/796
V26	5596	279211	19/967
V27	39163	245644	11/80
V28	30342	254465	75/704
Amount	31904	252903	41/366
Class	492	0	1

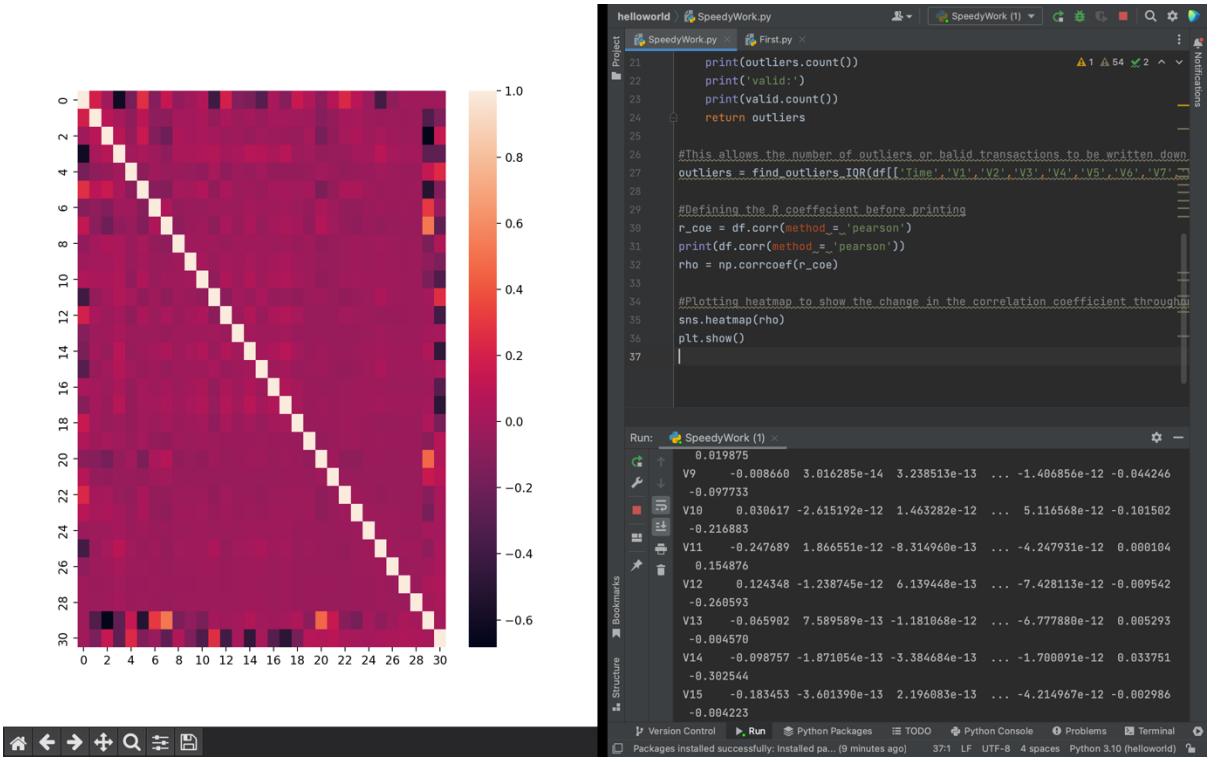
In order to obtain the outlier fractions per columns in the dataset, the number of fraudulent and valid transactions per column had to be ascertained first then tabulated. An outlier in statistics is defined as a point of data that falls more than 1.5 times the interquartile range above the third quartile or below the first quartile. These points on this dataset are more likely to be deemed as potential fraudulent and need to be flagged up. Using python, 'outliers' were defined and listed as shown in the screenshot above. Valid transactions were also printed so the total number of transactions could be identified to use to obtain the outlier fraction. Finally, the columns I wanted to show and find outliers for were listed individually.

Q3




```
#This allows the number of outliers or balid transactions to be written down per column name
outliers = find_outliers_IQR(df[['Time','V1','V2','V3','V4','V5','V6','V7','V8','V9','V10','V11','V12','V13','V14','V15','V16','V17','V18','V19','V20','V21','V22','V23
#Defining the R coeffecient before printing
r_coe = df.corr(method='pearson')
print(df.corr(method='pearson'))
rho = np.corrcoef(r_coe)

#Plotting scatter graphs to show the change in the correlation coefficient throughout the data
fig, ax = plt.subplots(nrows=1, ncols=3, figsize=(12, 3))
for i in [0,1,2]:
    ax[i].scatter(rho[0],rho[1+i])
    ax[i].title.set_text('Correlation = ' + "{:.2f}".format(rho[0,i+1]))
    ax[i].set(xlabel='rho',ylabel='y')
fig.subplots_adjust(wspace=.4)
plt.show()
```

```
dtype: int64
               Time            V1            V2  ...           V28    Amount     Class
Time       1.000000  1.173963e-01 -1.059333e-02  ... -9.412688e-03 -0.010596 -0.012323
V1         0.117396  1.000000e+00  3.777823e-12  ... -9.769215e-13 -0.227709 -0.101347
V2        -0.010593  3.777823e-12  1.000000e+00  ...  2.525513e-12 -0.531409  0.091289
V3        -0.419618 -2.118614e-12  2.325661e-12  ...  5.189109e-12 -0.210880 -0.192961
V4        -0.105260 -1.733159e-13 -2.314981e-12  ... -2.832372e-12  0.098732  0.133447
V5         0.173072 -3.473231e-12 -1.831952e-12  ...  1.0101196e-11 -0.386356 -0.094974
V6        -0.063016 -1.386165e-13  9.438444e-13  ... -6.069227e-13  0.215981 -0.043643
V7         0.084714 -1.116494e-13  5.403436e-12  ...  2.958679e-13  0.397311 -0.187257
V8        -0.036949  2.114527e-12  2.133785e-14  ...  1.866598e-12 -0.103079  0.019875
V9        -0.008660  3.016285e-14  3.238513e-13  ... -1.406856e-12 -0.044246 -0.097733
V10        0.030617 -2.615192e-12  1.463282e-12  ...  5.116568e-12 -0.101502 -0.216883
V11       -0.247689  1.866551e-12 -8.314960e-13  ... -4.247931e-12  0.000104  0.154876
V12        0.124348 -1.238745e-12  6.139448e-13  ... -7.428113e-12 -0.009542 -0.260593
```



These images depict the script and resulting plots or heatmap to show the correlation within the dataset. The R values change throughout the data, however, majority of the data show no correlation as the R is very close to 0. There are some points in the data where R changes are can reach as low as -0.6 or 0.6..