CMP5101 Data Mining

Project Report for Company Sales Data

In this project, I aimed at exploring the dataset “Company Sales” by using Data Mining techniques on PyCharm. For a better visualization of figures generated, I re-run the codes on Jupyter and gave place to them as they appear there. After exploring the basic statistical attributes of the data in hand, I applied linear regression to see the relationship between two columns in the dataframe: namely ‘Total Profit’ and ‘Product Profit’. Attached are the files containing the code and the data in .py and .csv formats respectively.

Importing the libraries and the dataset to be used:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics

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

Have a basic look at the data format:

```
print(data.head())
```

```
        Name Surname       ID  Total Profit  Product Profit  P1  P2  P3  
P4  \
0    Karoline Deason  1247791          85.0              80   5   2  12  
2
1        Xenia Crago  1906020          85.0              65   2  12   7  
2
2  France Buterbaugh  1900192          80.0              60  12  12  12  
2
3      Lottie Gryder  1907020          85.0              65   7  12  12  
2
4      Corie Woodley  1905678          80.0              85   7  12  12  
2

   P5  P6  ...  P11  P12  P13  P14  P15  P16  P17  P18  P19  P20
0   2  12  ...    5    2    5    7    7   12    5    7    2    7
1   2  12  ...   12    2    5    7    7    7    7    5    2    7
2   5   2  ...    7    2    5    7    7    7    2    5    2    7
3   5   2  ...    7    2    5    7    7   12   12    5    2    5
4   2  12  ...    5    2   12    7    7   12    7   12    5    5
```

Info about our dataset:

```
print(data.info())
```

```
Name Surname      53 non-null object
ID                53 non-null object
Total Profit      47 non-null float64
Product Profit    53 non-null int64
P1                53 non-null int64
P2                53 non-null int64
P3                53 non-null int64
P4                53 non-null int64
P5                53 non-null int64
P6                53 non-null int64
P7                53 non-null int64
P8                53 non-null int64
P9                53 non-null int64
P10               53 non-null int64
P11               53 non-null int64
P12               53 non-null int64
P13               53 non-null int64
P14               53 non-null int64
P15               53 non-null int64
P16               53 non-null int64
P17               53 non-null int64
P18               53 non-null int64
P19               53 non-null int64
P20               53 non-null int64
dtypes: float64(1), int64(21), object(2)
memory usage: 10.1+ KB
```

Apparently, there are missing values in ‘Total Profit’ column.

```
print(data.isnull().sum())
```

Exactly 6 of them.

So, we fill those missing values with mean value of that column via the following code:

```
data.fillna(data.mean(), inplace=True)
```

Then, we acquire the basic statistical information about our dataset:

```
print(data.describe())
```

```
       Total Profit  Product Profit         P1         P2         P3  \
count     53.000000       53.000000  53.000000  53.000000  53.000000
mean      85.638298       76.603774   7.377358   9.433962  11.528302
std        7.443184       10.997097   3.045971   3.905171   1.475489
min       70.000000       50.000000   2.000000   2.000000   7.000000
25%       80.000000       70.000000   5.000000   5.000000  12.000000
50%       85.000000       80.000000   7.000000  12.000000  12.000000
75%       85.638298       85.000000   7.000000  12.000000  12.000000
max      110.000000      100.000000  12.000000  12.000000  12.000000
```

Visual exploration of the dataset:

```
print(data.boxplot(column=['Total Profit', 'Product Profit']))
```

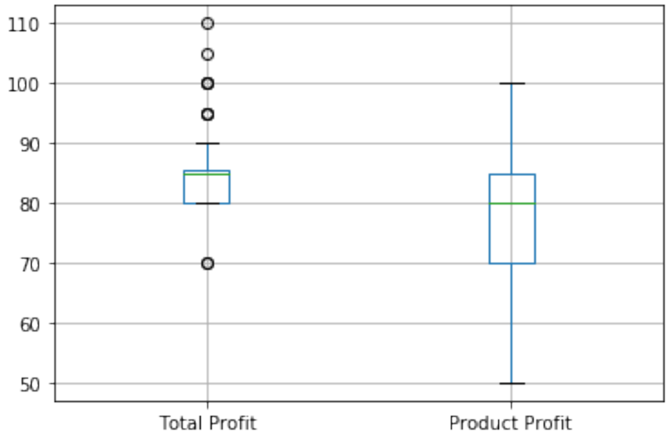
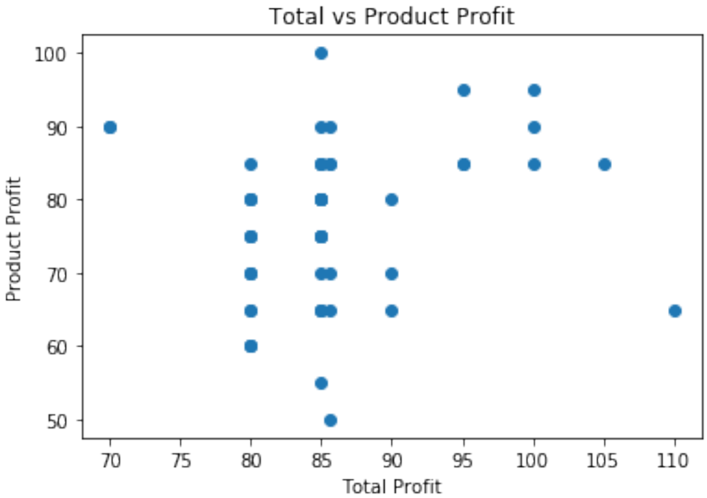


Figure above shows that there are no outliers in Product Profit attribute, however, in Total Profit column, we have some outliers.

For seeing the relation between two attributes in a visual format, we create a scatter plot as following:

```
tp = data['Total Profit']
pp = data['Product Profit']

plt.scatter(tp, pp)
plt.title('Total vs Product Profit')
plt.xlabel('Total Profit')
plt.ylabel('Product Profit')
plt.show()
```

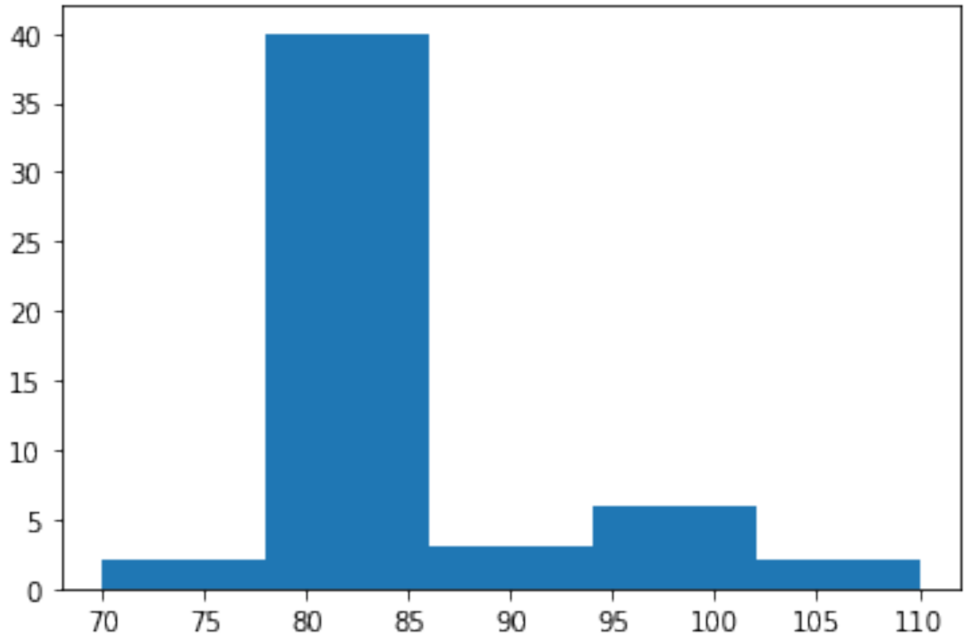


Here, we see that our mean value of Total Profit which replaced the missing values in our dataset is between 85 and 90 for all the other values given in our dataframe are product of number 5. Also, if we wanted to make a cluster analysis there would be three outliers according to this figure: the ones with values 70, 90; 85,100, and 110,65 as Total and Product Profit values respectively.

Below, we see how Total and Product Profit values are distributed in the form of histograms:

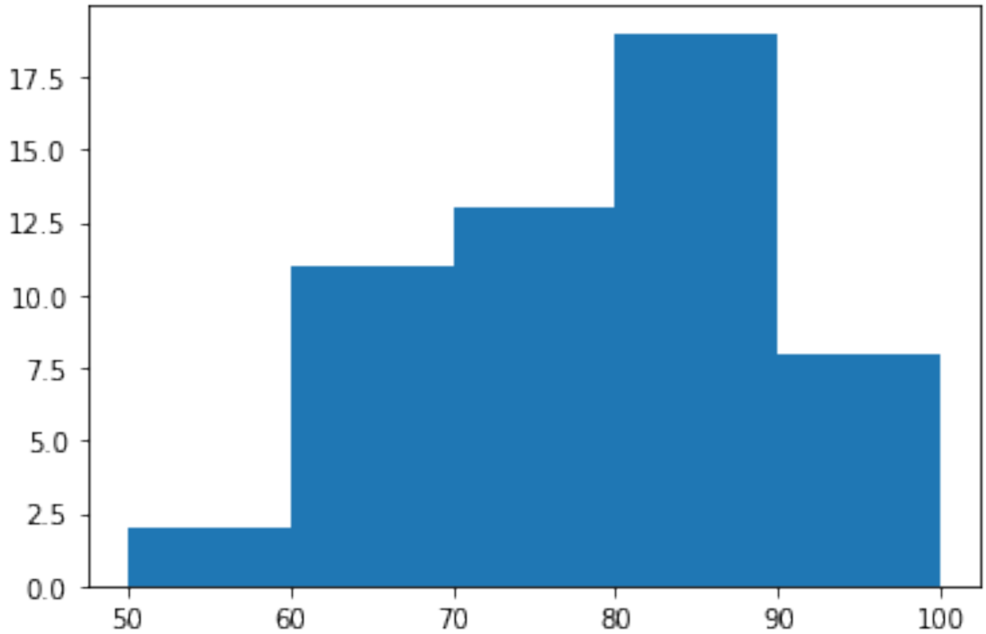
Total Profit:

```
plt.hist(tp, bins=5)
plt.show()
```



Product Profit:

```
plt.hist(pp, bins=5)
plt.show()
```



Calculating the means for each product and sorting them:

```
product_means = data.iloc[:, 4:].mean(axis=0)
print(product_means.sort_values(ascending=False))
```

```
P3     11.528302
P16    10.867925
P8     10.547170
P2      9.433962
P6      8.811321
P10     7.377358
P1      7.377358
P15     7.094340
P14     7.018868
P17     6.339623
P18     6.150943
P11     6.075472
P20     5.528302
P13     5.358491
P7      3.207547
P5      3.207547
P4      2.660377
P9      2.547170
P12     2.528302
P19     2.113208
```

According to the series above, we see that the most sold item is P3 with a mean value of 11.5, whereas P19 is the least sold one with 2.1 mean value.

Generating a linear regression model for our dataset by splitting it into train and test dataset with 80% and 20% ratios accordingly:

```
X = data['Total Profit'].values.reshape(-1,1)
y = data['Product Profit'].values.reshape(-1,1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=0)
regressor = LinearRegression()
regressor.fit(X_train, y_train)
print(regressor.intercept_)
print(regressor.coef_)


plt.scatter(X_test, y_test)
plt.plot(X_test, y_pred, linewidth=2)
plt.show()


y_pred = regressor.predict(X_test)
actual_vs_predicted_result = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted':
y_pred.flatten()})
vis_of_AvsPR = actual_vs_predicted_result.head()
vis_of_AvsPR.plot(kind='bar',figsize=(16,10))
plt.grid(which='minor', linestyle='-', linewidth='0.5')
plt.grid(which='major', linestyle=':', linewidth='0.5')
plt.show()
print('Mean Absolute Error is', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error is', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error is', np.sqrt(metrics.mean_squared_error(y_test,
y_pred)))
```

```
[45.78836133]
[[0.35505068]]
```

In the end, we find a regression line with the intercept value of approximately 45.8 and a coefficient of 0.36, thus, the formula of our regression line and the figure of it are as below:

Product Profit = 45.78836133 + 0.35505068*(Total Profit)

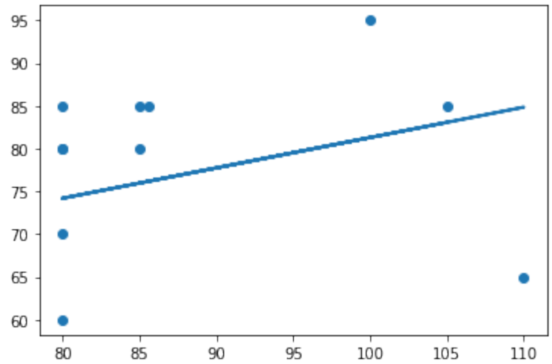
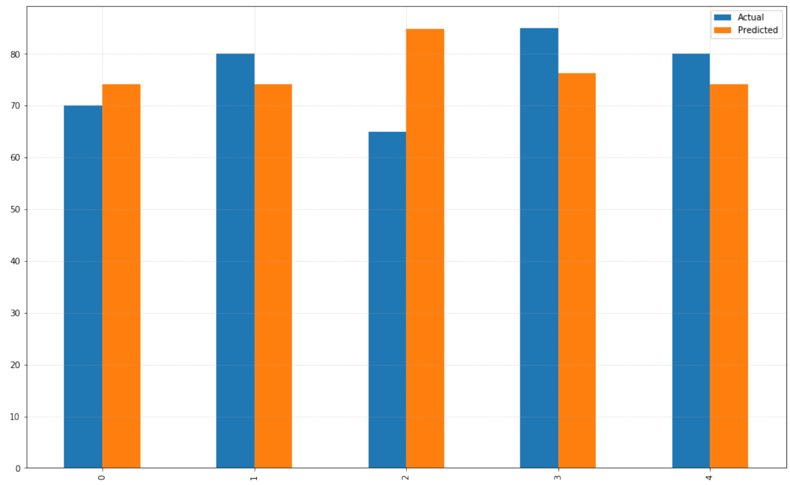


Figure below shows first 5 (head) presentations of the predicted and actual values for our test dataset:



Statistical attributions of our linear regression are as follows:

```
print('Mean Absolute Error is', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error is', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error is', np.sqrt(metrics.mean_squared_error(y_test,
y_pred)))
```

```
Mean Absolute Error is 8.923615610281765
Mean Squared Error is 105.82053773054382
Root Mean Squared Error is 10.286910990698026
```