Assignment 07

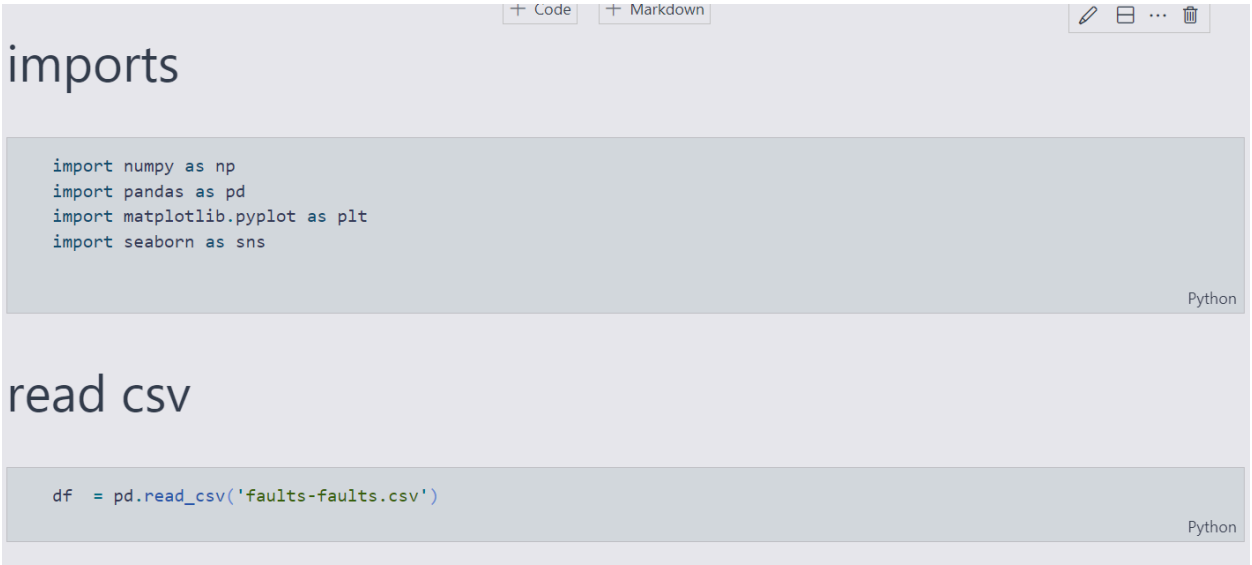
-by aditya verma

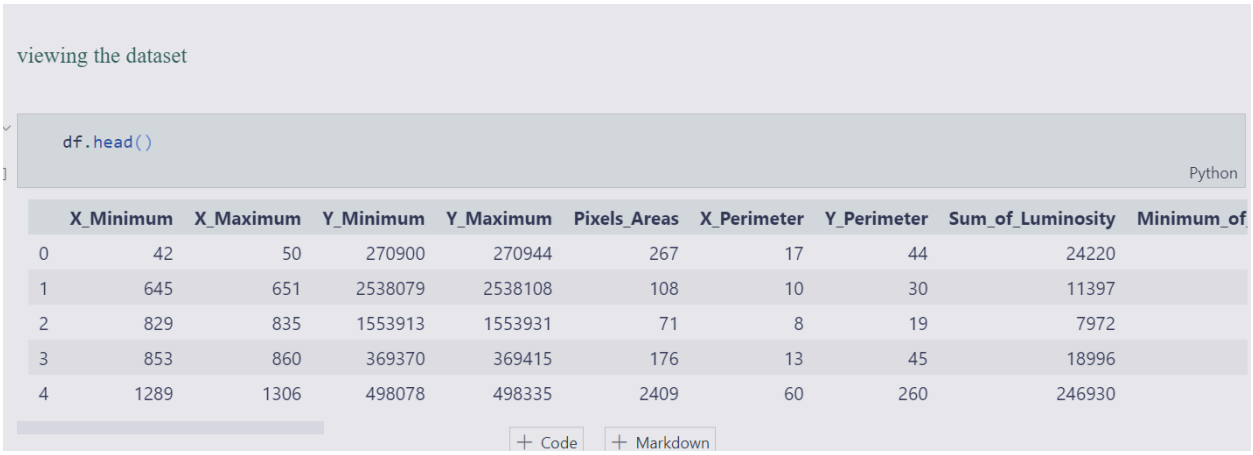
Dataset

https://drive.google.com/file/d/1INLNHDGhvnhgcFCDxSQzU_pNIMvAukiY/view?usp=classroom_web&authuser=0

Aim

Assignment-7 : Use KNN, SVM, and Decision Tree classifier to predict the steel plate fault prection and compare it in all benchmark



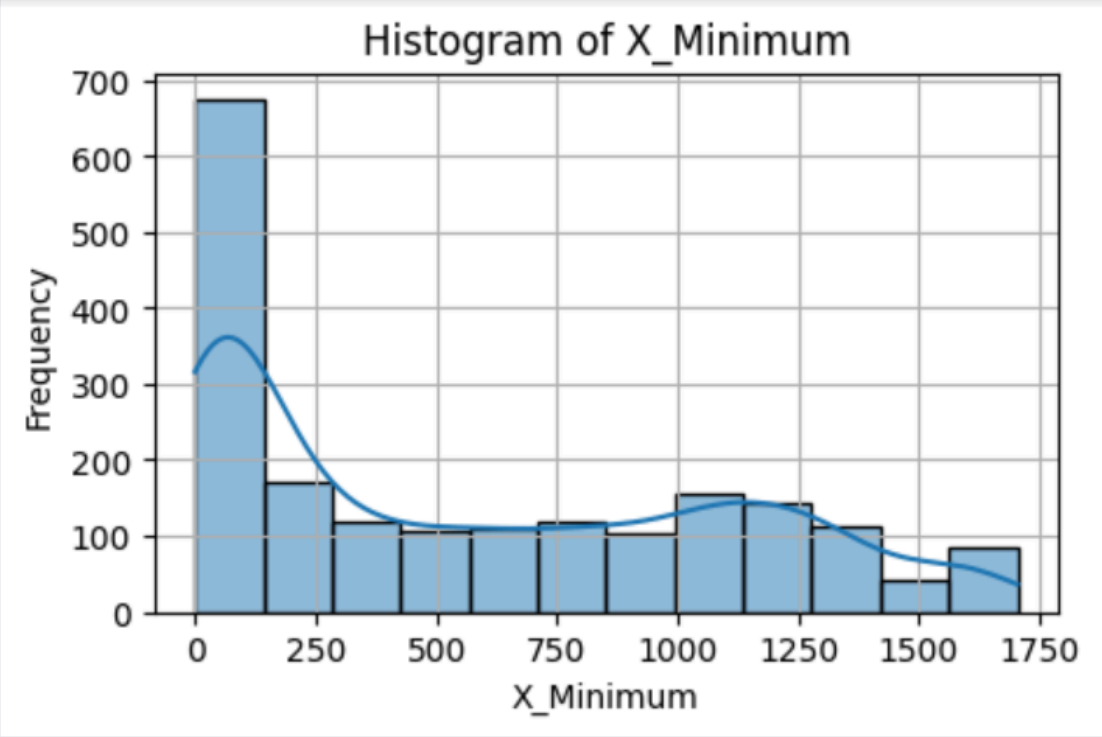


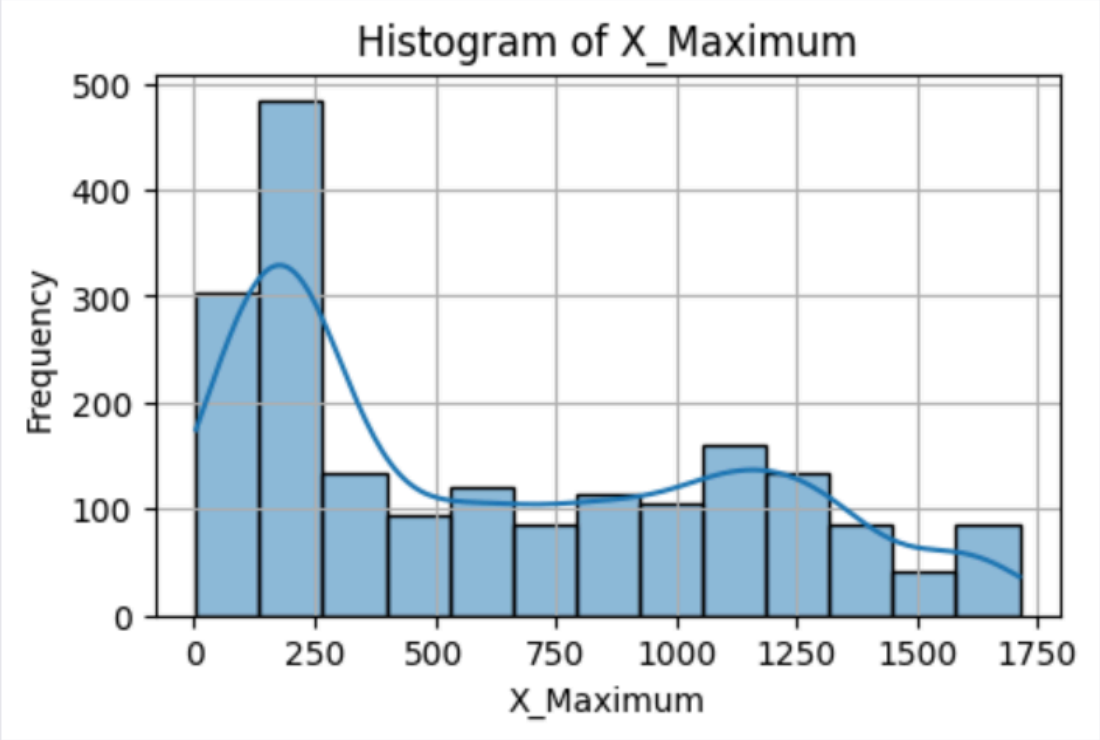
	X_Minimum	X_Maximum	Y_Minimum	Y_Maximum	Pixels_Areas	X_Perimeter	Y_Perimeter	Sum_of_Luminosity	Minimum_of
0	42	50	270900	270944	267	17	44	24220	
1	645	651	2538079	2538108	108	10	30	11397	
2	829	835	1553913	1553931	71	8	19	7972	
3	853	860	369370	369415	176	13	45	18996	
4	1289	1306	498078	498335	2409	60	260	246930	

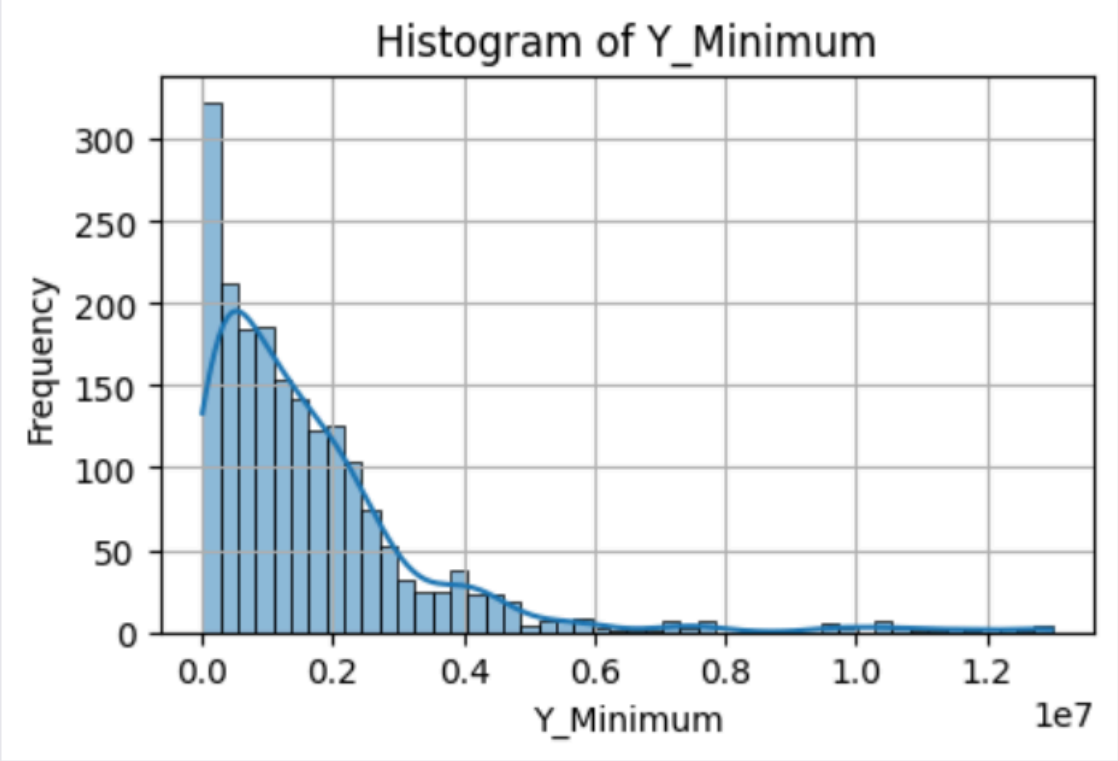
plotting the histogram for all the columns

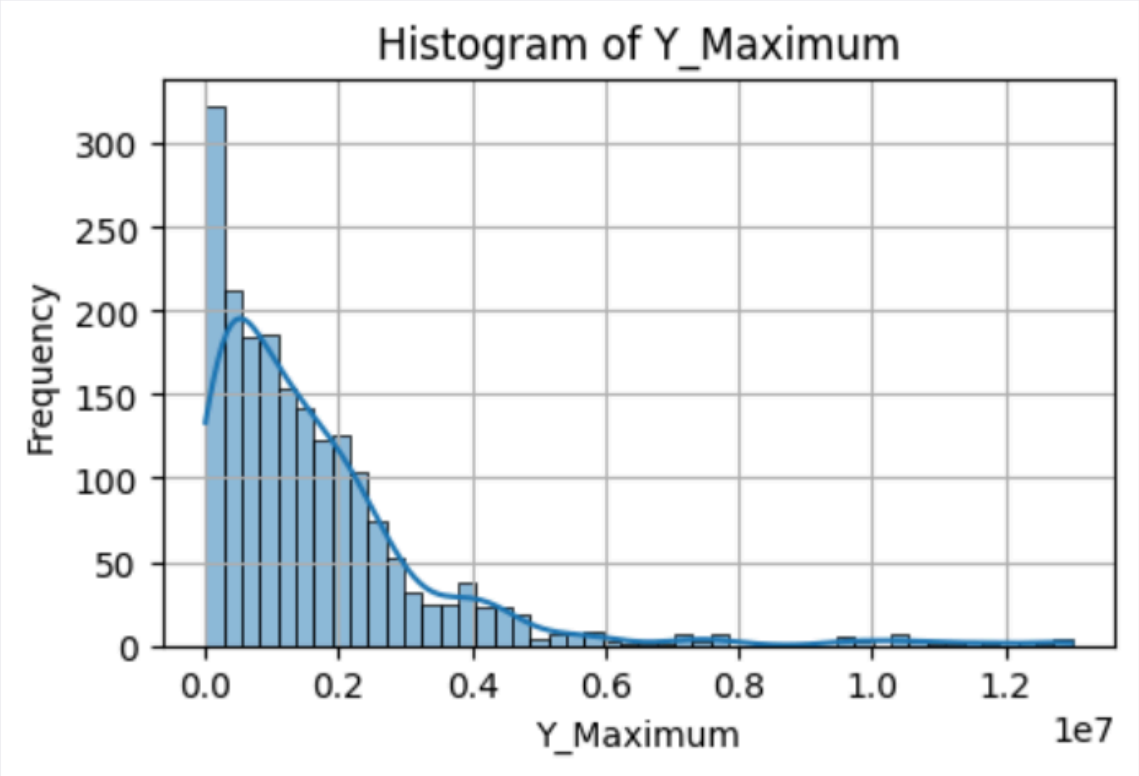
```
#plot histogram for all columns
for col in df.columns:
    plt.figure(figsize=(5, 3))
    sns.histplot(df[col], kde=True)
    plt.title(f'Histogram of {col}')
    plt.xlabel(col)
    plt.ylabel('Frequency')
    plt.grid()
    plt.show()
```

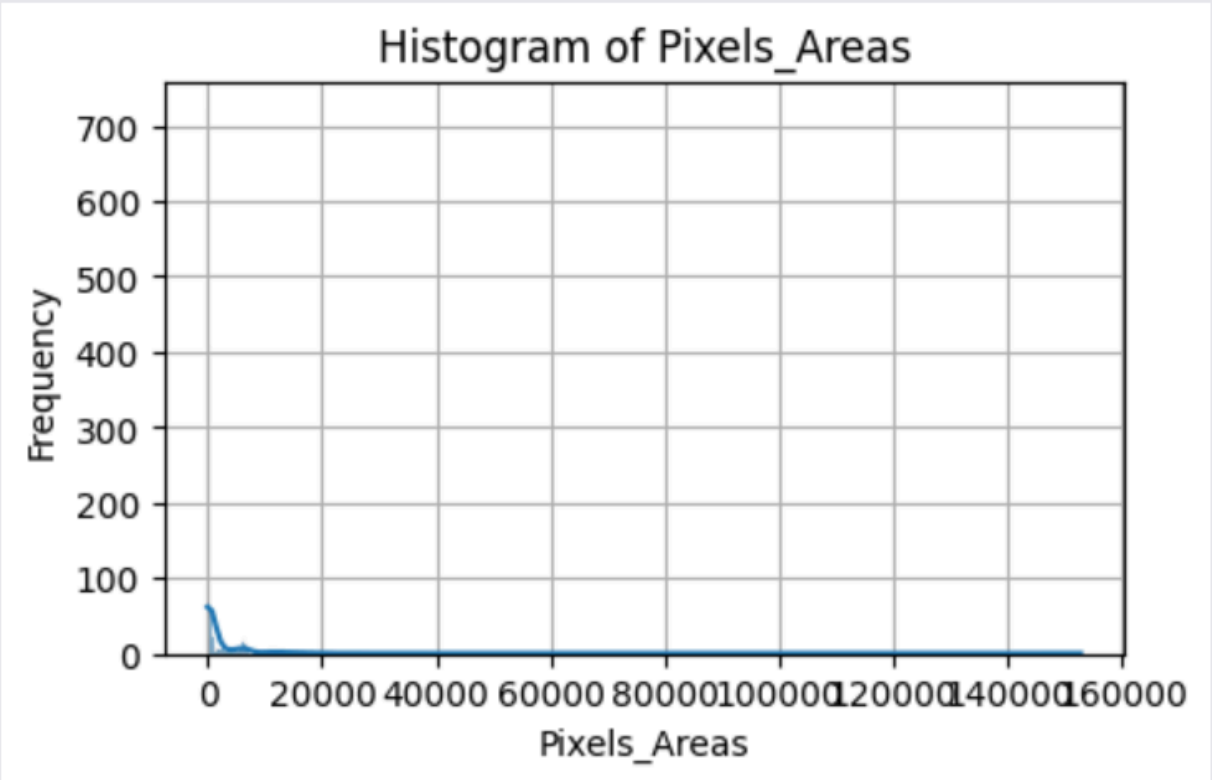
Pytho

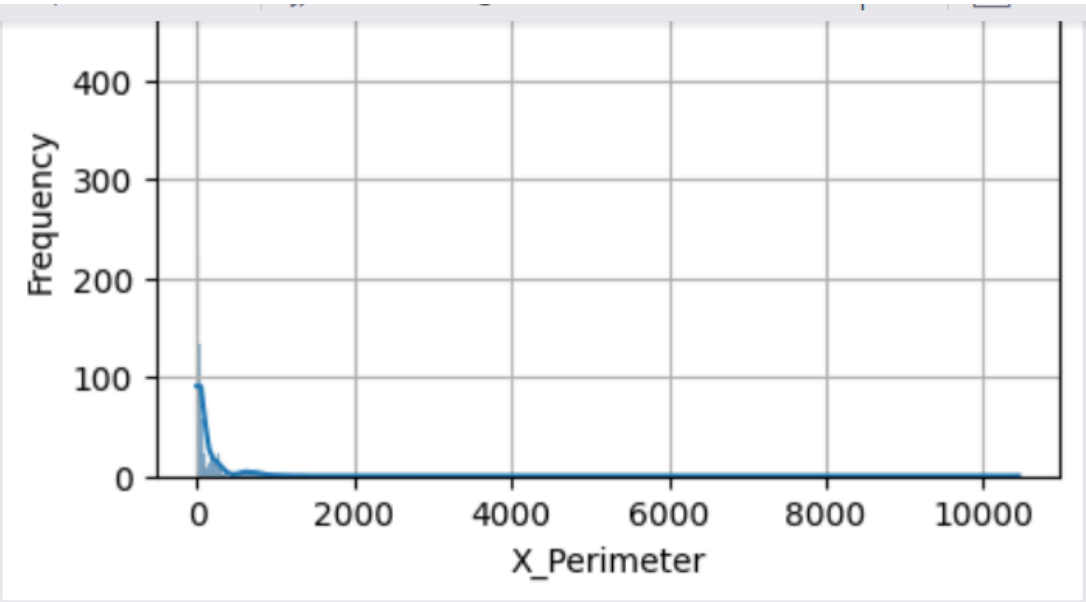


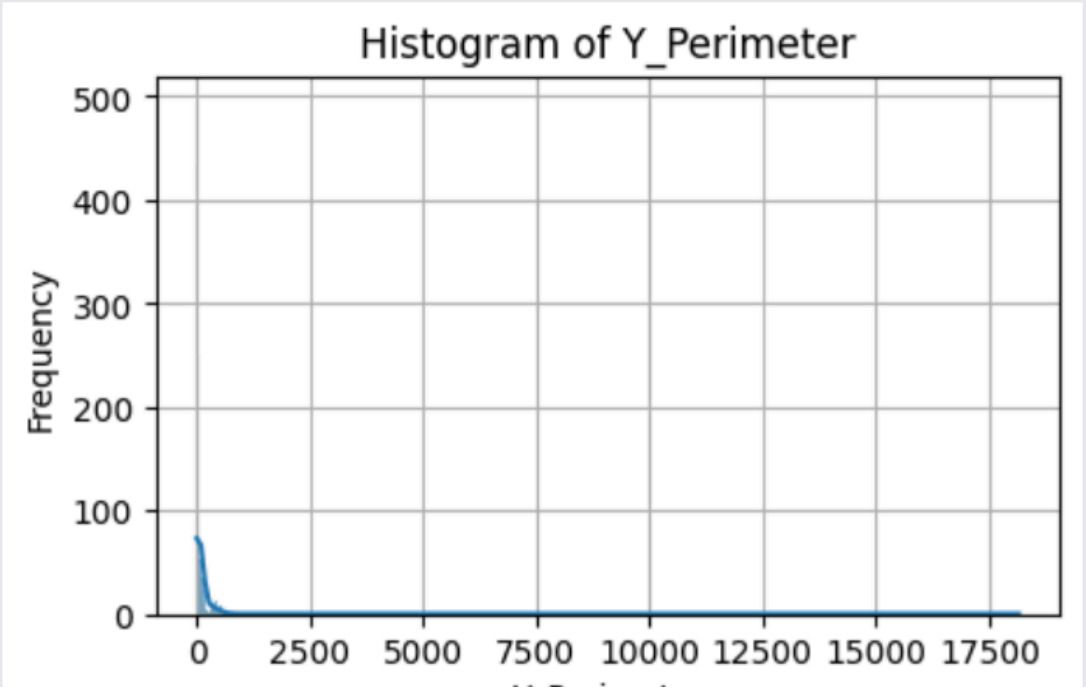

Histogram of X_Maximum
Frequency
X_Maximum
0
100
200
300
400
500
0
250
500
750
1000
1250
1500
1750

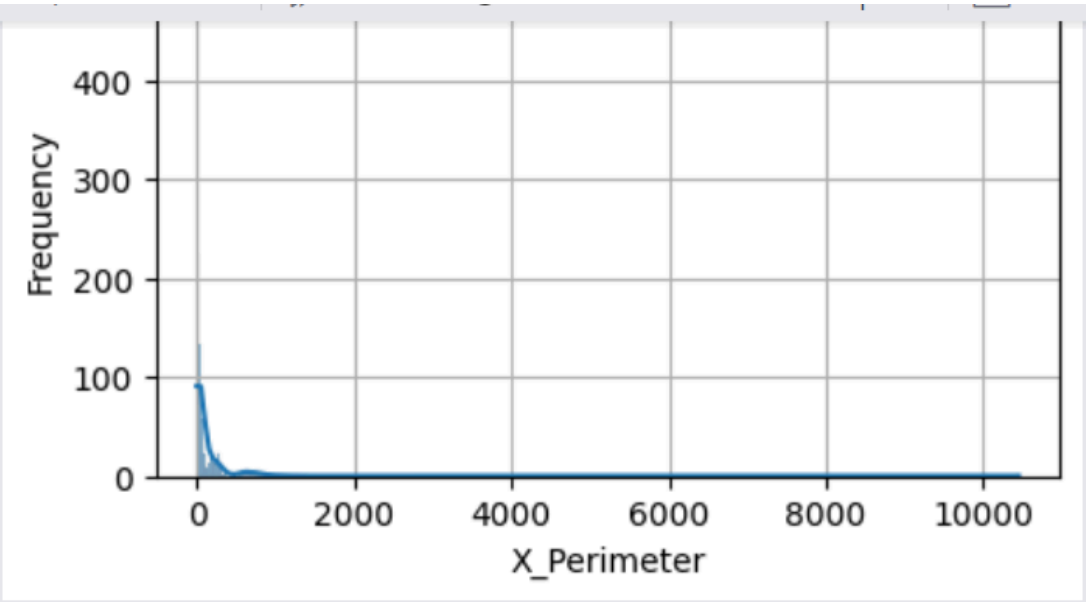

Histogram of Y_Minimum
Frequency
Y_Minimum
1e7
0
50
100
150
200
250
300
0.0
0.2
0.4
0.6
0.8
1.0
1.2

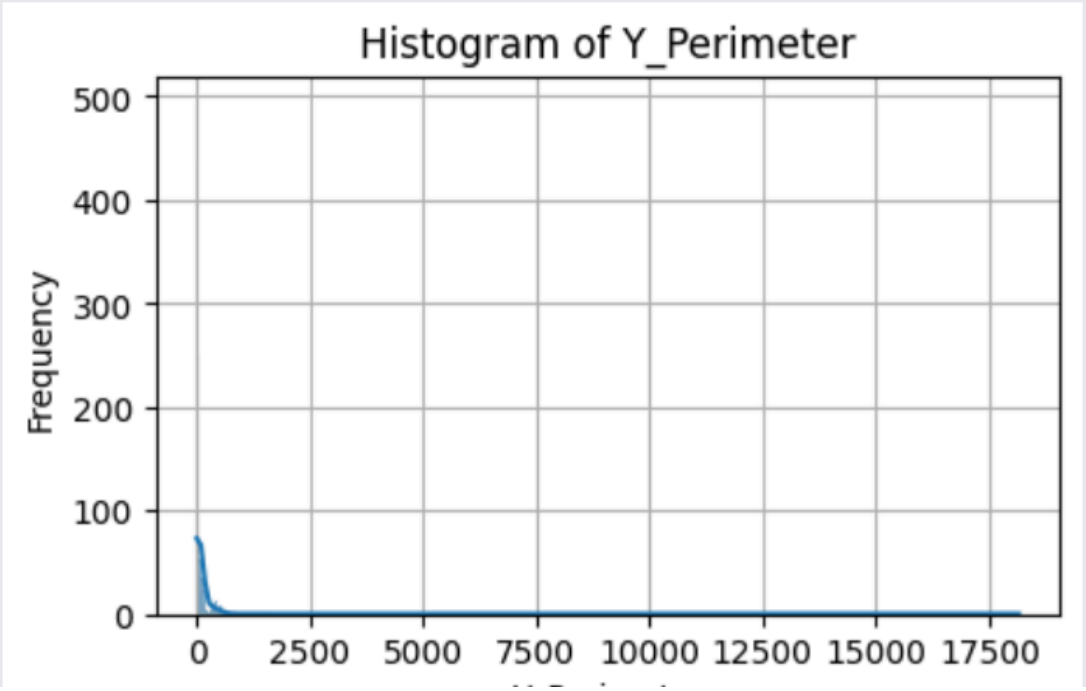

Histogram of Y_Maximum
Frequency
300
250
200
150
100
50
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
Y_Maximum
1e7

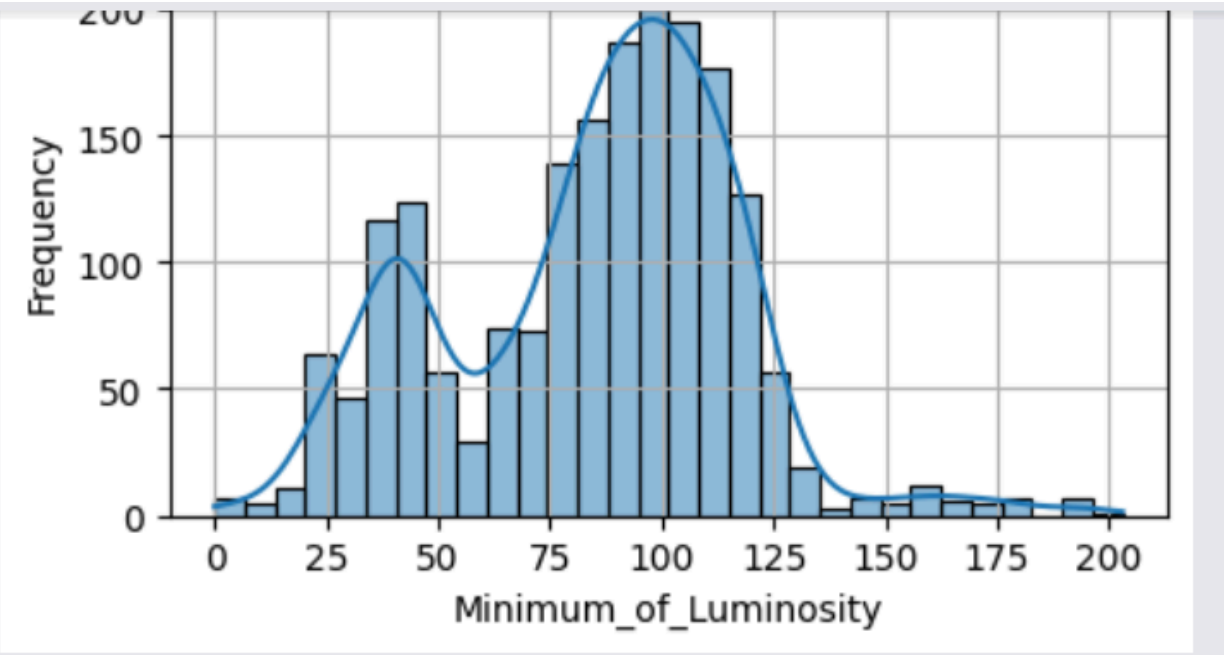

Histogram of Pixels_Areas
Frequency
700
600
500
400
300
200
100
0
0
20000
40000
60000
80000
100000
120000
140000
160000
Pixels_Areas

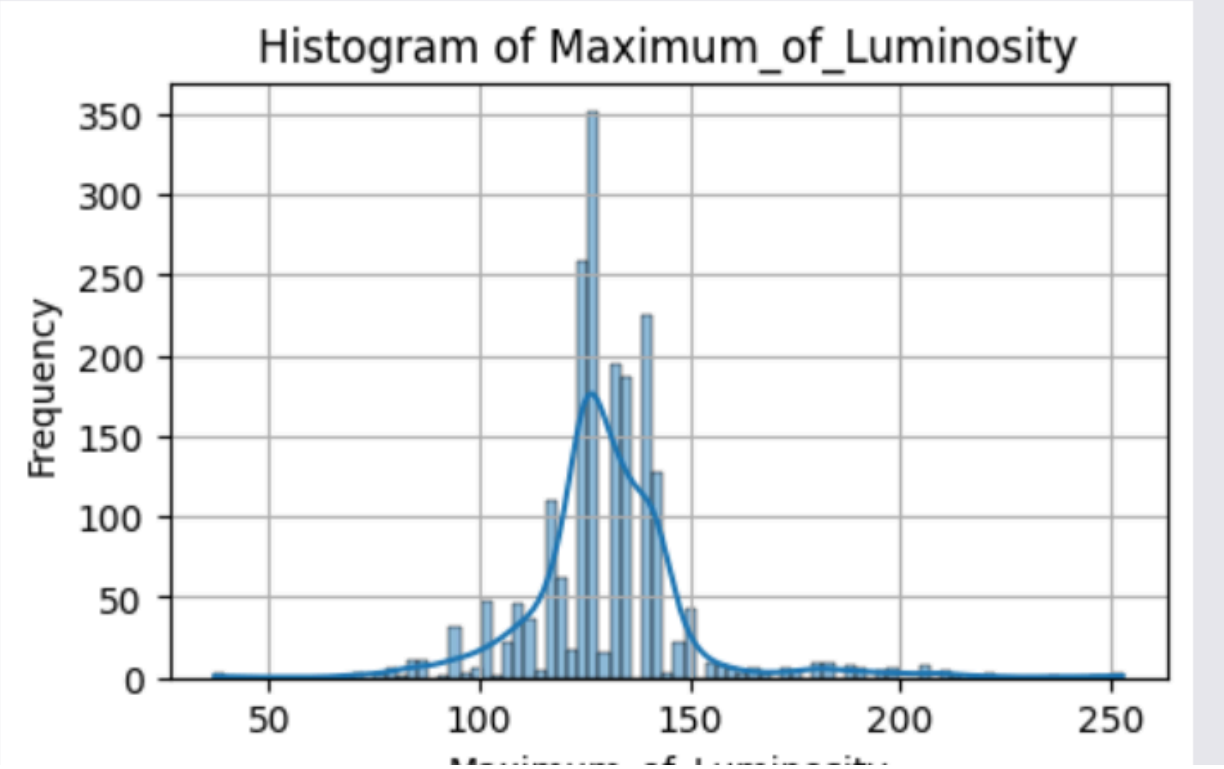

400
300
200
100
0
Frequency
0
2000
4000
6000
8000
10000
X_Perimeter


Histogram of Y_Perimeter
500
400
300
200
100
0
Frequency
0
2500
5000
7500
10000
12500
15000
17500


400
300
200
100
0
Frequency
0
2000
4000
6000
8000
10000
X_Perimeter


Histogram of Y_Perimeter
500
400
300
200
100
0
Frequency
0
2500
5000
7500
10000
12500
15000
17500


Frequency
150
100
50
0
0
25
50
75
100
125
150
175
200
Minimum_of_Luminosity


Histogram of Maximum_of_Luminosity
Frequency
350
300
250
200
150
100
50
0
50
100
150
200
250

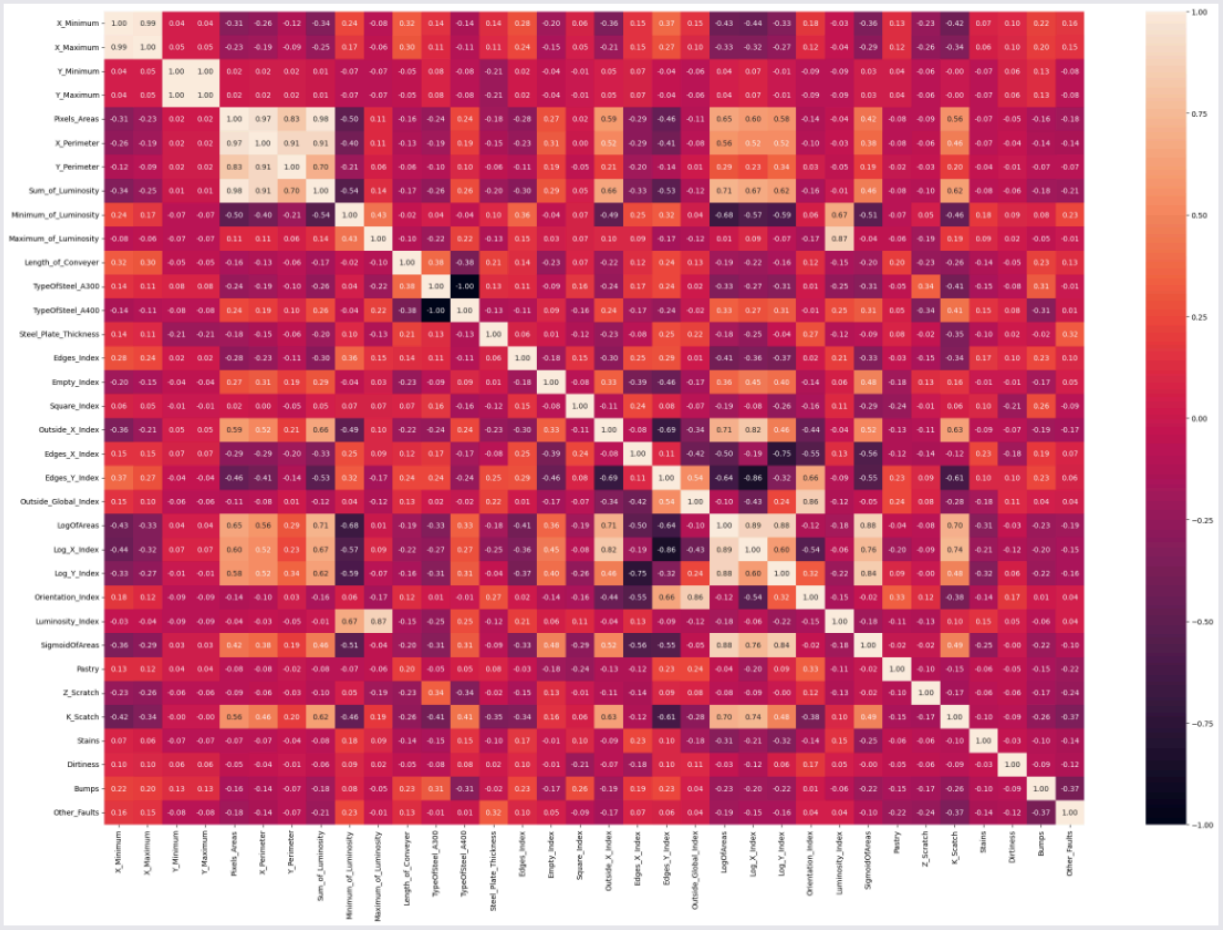
plotting the coorelation heatmap of the dataset

```
crorrelation_matrix = df.corr()
plt.figure(figsize=(30, 20))
sns.heatmap(crorrelation_matrix, annot=True, fmt='.2f')
```

Python

6]

<Axes: >




```
# seprating the columns for x and y set
```

markdown

```
X = df.drop(columns=['Pastry','Z_Scratch','K_Scatch','Stains','Dirtiness','Bumps','Other_Faults'])
y = df[['Pastry','Z_Scratch','K_Scatch','Stains','Dirtiness','Bumps','Other_Faults']]
```

Python

```
X
```

Python

merging the columns of y data frame to one classifying column

```
y =[]
for i in range(faults.shape[0]):
    if faults.iloc[i]['Pastry'] == 1:
        y.append('Pastry')
    elif faults.iloc[i]['Z_Scratch'] == 1:
        y.append('Z_Scratch')
    elif faults.iloc[i]['K_Scatch'] == 1:
        y.append('K_Scatch')
    elif faults.iloc[i]['Stains'] == 1:
        y.append('Stains')
    elif faults.iloc[i]['Dirtiness'] == 1:
        y.append('Dirtiness')
    elif faults.iloc[i]['Bumps'] == 1:
        y.append('Bumps')
    else:
        y.append('Other_Faults')
```

```
y= np.array(y)
y
```

Python

```
array(['Pastry', 'Pastry', 'Pastry', ..., 'Other_Faults', 'Other_Faults',
       'Other_Faults'], shape=(1941,), dtype='<U12')
```

Splitting the data sets for training and test set

+ Code + Markdown Add Markdown Cell

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Python

CREATING THE PIPELINE FOR THE VARIOUS MODELS TO TEST UPON

```
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

#define classification models
models ={
    'LogisticRegression': LogisticRegression(max_iter=1000),
    'DecisionTreeClassifier':DecisionTreeClassifier(),
    'RandomForestClassifier': RandomForestClassifier(),
    'svc': SVC(),
    'KNeighborsClassifier': KNeighborsClassifier(n_neighbors=5),

}
#train and evaluate each model
for name,model in models.items():
    print(f'Training {name}...')
    model.fit(X_train,y_train)
    y_predict =model.predict(X_test)
    accuracy = accuracy_score(y_test, y_predict)
    print(f'Accuracy of {name}: { accuracy * 100:.2f}%')
    print(classification_report(y_test,y_predict))

    print('-' * 50)
    print('Confusion Matrix:')
```

```
    print('-' * 50)
    print('Confusion Matrix:')
    confusion = confusion_matrix(y_test,y_predict)
    print(confusion)

    print('-' * 50)
    print('-' * 50)
```

```
# supreess warnings
from sklearn.exceptions import ConvergenceWarning
import warnings
warnings.filterwarnings("ignore", category=ConvergenceWarning)
```

Python

```
Training LogisticRegression...
Accuracy of LogisticRegression: 52.44%
              precision    recall  f1-score   support

       Bumps       0.53      0.24      0.33        72
   Dirtiness       0.00      0.00      0.00         8
    K_Scatch       0.91      0.75      0.82        83
Other_Faults       0.44      0.87      0.58       143
      Pastry       0.00      0.00      0.00        29
      Stains       0.00      0.00      0.00        13
   Z_Scratch       0.00      0.00      0.00        41

    accuracy                           0.52       389
   macro avg       0.27      0.27      0.25       389
weighted avg       0.45      0.52      0.45       389


--------------------------------------------------
Confusion Matrix:
[[ 17   0   0  55   0   0   0]
 [  1   0   0   7   0   0   0]
 [  0   0  62  19   0   2   0]
 [ 12   0   5 125   0   1   0]
 [  2   0   1  26   0   0   0]
 [  0   0   0  13   0   0   0]
 [  0   0   0  41   0   0   0]]
...
 [ 5  1  0  5  0  0  2]
 [16  1  0 22  0  0  2]]
--------------------------------------------------
--------------------------------------------------
```

```
Training DecisionTreeClassifier...
Accuracy of DecisionTreeClassifier: 73.78%
              precision    recall  f1-score   support

       Bumps       0.54      0.60      0.57        72
   Dirtiness       0.75      0.75      0.75         8
    K_Scatch       0.94      0.93      0.93        83
Other_Faults       0.77      0.69      0.73       143
      Pastry       0.37      0.45      0.41        29
      Stains       0.86      0.92      0.89        13
   Z_Scratch       0.88      0.90      0.89        41

    accuracy                           0.74       389
   macro avg       0.73      0.75      0.74       389
weighted avg       0.75      0.74      0.74       389


--------------------------------------------------
Confusion Matrix:
[[43  1  1 19  8  0  0]
 [ 1  6  0  1  0  0  0]
 [ 2  0 77  1  2  0  1]
 [25  0  2 99 11  2  4]
 [ 7  1  0  8 13  0  0]
 [ 1  0  0  0  0 12  0]
 [ 0  0  2  1  1  0 37]]
--------------------------------------------------
```

```
Training RandomForestClassifier...
Accuracy of RandomForestClassifier: 77.63%
              precision    recall  f1-score   support

       Bumps       0.62      0.65      0.64        72
   Dirtiness       0.75      0.75      0.75         8
    K_Scatch       0.96      0.93      0.94        83
Other_Faults       0.73      0.77      0.75       143
      Pastry       0.54      0.45      0.49        29
      Stains       1.00      0.92      0.96        13
   Z_Scratch       0.95      0.90      0.93        41

    accuracy                           0.78       389
   macro avg       0.79      0.77      0.78       389
weighted avg       0.78      0.78      0.78       389

-------------------------------------------------------
Confusion Matrix:
[[ 47   0   0  20   5   0   0]
 [  0   6   0   2   0   0   0]
 [  0   0  77   6   0   0   0]
 [ 21   2   2 110   6   0   2]
 [  6   0   0  10  13   0   0]
 [  1   0   0   0   0  12   0]
 [  1   0   1   2   0   0  37]]
-------------------------------------------------------
```

```
Training svc...
Accuracy of svc: 51.67%
              precision    recall  f1-score   support

       Bumps       0.41      0.19      0.26        72
   Dirtiness       0.00      0.00      0.00         8
    K_Scatch       0.91      0.73      0.81        83
Other_Faults       0.44      0.88      0.59       143
      Pastry       0.00      0.00      0.00        29
      Stains       0.00      0.00      0.00        13
   Z_Scratch       0.00      0.00      0.00        41

    accuracy                           0.52       389
   macro avg       0.25      0.26      0.24       389
weighted avg       0.43      0.52      0.44       389

-------------------------------------------------------
Confusion Matrix:
[[ 14   0   2  54   2   0   0]
 [  1   0   0   7   0   0   0]
 [  1   0  61  21   0   0   0]
 [ 14   0   3 126   0   0   0]
 [  4   0   1  24   0   0   0]
 [  0   0   0  13   0   0   0]
 [  0   0   0  41   0   0   0]]
```

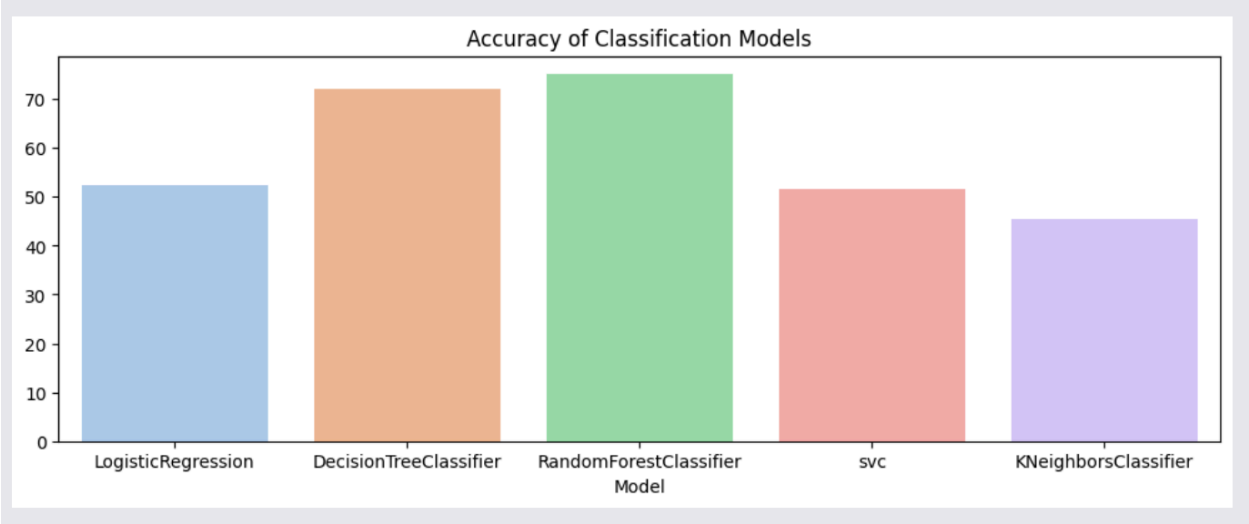
```
----------------------------------------------------
Training KNeighborsClassifier...
Accuracy of KNeighborsClassifier: 45.50%
              precision    recall  f1-score   support

      Bumps       0.23      0.36      0.28        72
  Dirtiness       0.38      0.38      0.38         8
   K_Scatch       0.88      0.77      0.82        83
Other_Faults       0.50      0.57      0.53       143
     Pastry       0.08      0.03      0.05        29
     Stains       0.00      0.00      0.00        13
  Z_Scratch       0.12      0.05      0.07        41

   accuracy                           0.46       389
  macro avg       0.31      0.31      0.30       389
weighted avg       0.44      0.46      0.44       389

----------------------------------------------------
Confusion Matrix:
[[26  2  1 32  4  2  5]
 [ 4  3  0  0  0  0  1]
 [ 8  0 64  7  0  0  4]
 [43  1  6 81  7  2  3]
 [11  0  2 14  1  1  0]
 [ 5  1  0  5  0  0  2]
 [16  1  0 22  0  0  2]]
----------------------------------------------------
```

comparision of the various outcomes

```
#plot the bar graph figure for the accuracy of each model
accuracy_list = []
for name, model in models.items():
    model.fit(X_train, y_train)
    y_predict = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_predict)
    accuracy_list.append(accuracy * 100)
plt.figure(figsize=(12, 4))
sns.barplot(x=list(models.keys()), y=accuracy_list, palette='pastel')
plt.title('Accuracy of Classification Models')
plt.xlabel('Model')
```

Pytho

```
Text(0.5, 0, 'Model')
```



Conclusion

Colab link

https://colab.research.google.com/drive/1kKHViBDT0fU5UWzbcZCWa6fvrXpE7VCa?usp=sharing