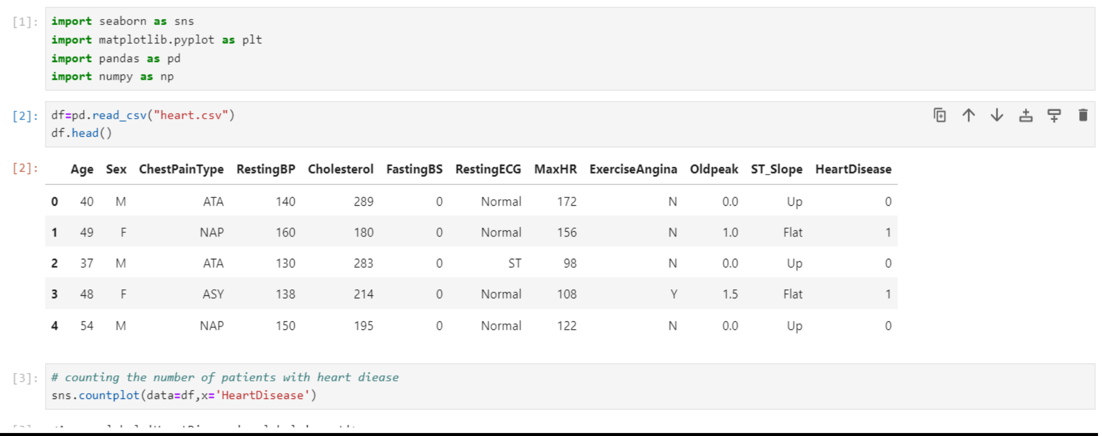
Heart Diseases prediction using KNN Algorithm

- Aim: To create a KNN model and use the training and split ratio to train the model
- STEP 1 : The training features include the age ,sex, chest pain type, resting bp ,cholesterol ,Fastingbs , resting ecg, Maxhr, exercise angenia, old peak and st slope
- The head method of pandas is used to give a brief overview of the data set



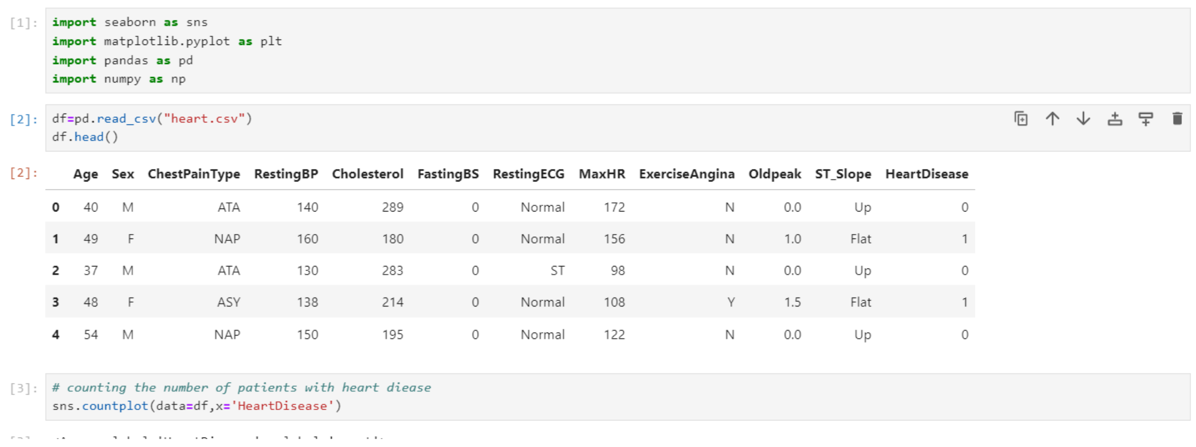
```
[1]: import seaborn as sns
     import matplotlib.pyplot as plt
     import pandas as pd
     import numpy as np
```

```
[2]: df=pd.read_csv("heart.csv")
     df.head()
```

```
[2]:    Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease
     0   40    M            ATA        140          289          0      Normal    172               N      0.0        Up             0
     1   49    F            NAP        160          180          0      Normal    156               N      1.0      Flat             1
     2   37    M            ATA        130          283          0          ST     98               N      0.0        Up             0
     3   48    F            ASY        138          214          0      Normal    108               Y      1.5      Flat             1
     4   54    M            NAP        150          195          0      Normal    122               N      0.0        Up             0
```

```
[3]: # counting the number of patients with heart diease
     sns.countplot(data=df,x='HeartDisease')
```

- STEP 2: Seaborn visualization(Countplot) is used to give the count of heart diseases patients suffering the diseases or not suffering. Additionally, the hot encoding is used using the label encoder class for converting the categorical data into numerical format. The correlation method shows the relationship between the different features and the heart diseases



```
[1]: import seaborn as sns
     import matplotlib.pyplot as plt
     import pandas as pd
     import numpy as np
```

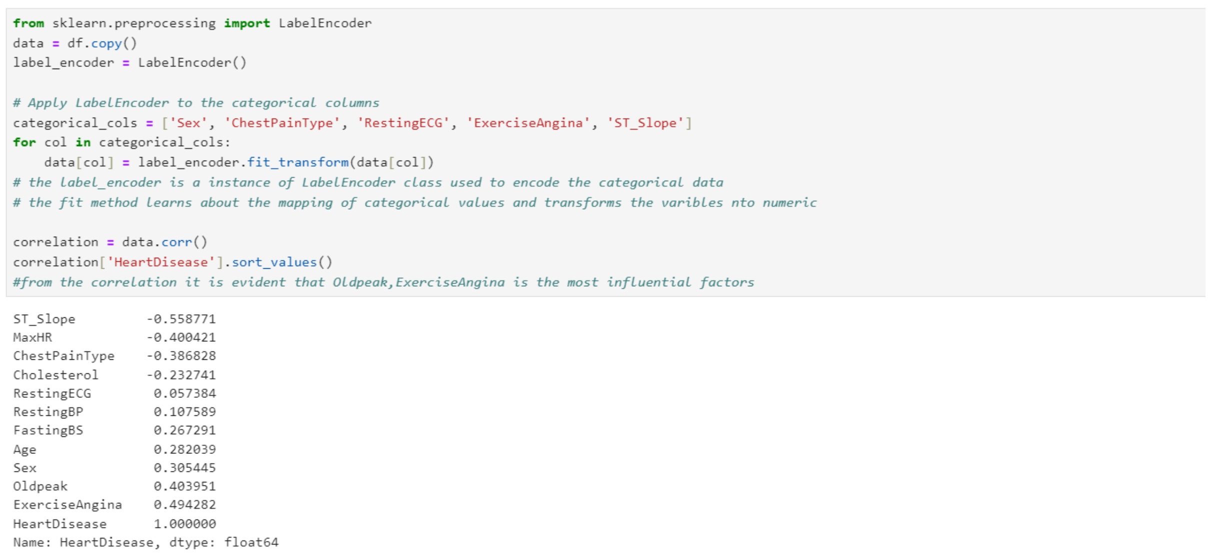
```
[2]: df=pd.read_csv("heart.csv")
     df.head()
```

[2]:

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	0	Normal	172	N	0.0	Up	0
1	49	F	NAP	160	180	0	Normal	156	N	1.0	Flat	1
2	37	M	ATA	130	283	0	ST	98	N	0.0	Up	0
3	48	F	ASY	138	214	0	Normal	108	Y	1.5	Flat	1
4	54	M	NAP	150	195	0	Normal	122	N	0.0	Up	0

```
[3]: # counting the number of patients with heart diease
     sns.countplot(data=df,x='HeartDisease')
```

- STEP 3: Exercise Angenia and the old peak are the major contributors of the heart diseases



```
from sklearn.preprocessing import LabelEncoder
data = df.copy()
label_encoder = LabelEncoder()

# Apply LabelEncoder to the categorical columns
categorical_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
for col in categorical_cols:
    data[col] = label_encoder.fit_transform(data[col])
# the label_encoder is a instance of LabelEncoder class used to encode the categorical data
# the fit method learns about the mapping of categorical values and transforms the varibles nto numeric

correlation = data.corr()
correlation['HeartDisease'].sort_values()
#from the correlation it is evident that Oldpeak,ExerciseAngina is the most influential factors
```

```
ST_Slope         -0.558771
MaxHR            -0.400421
ChestPainType    -0.386828
Cholesterol      -0.232741
RestingECG        0.057384
RestingBP         0.107589
FastingBS         0.267291
Age               0.282039
Sex               0.305445
Oldpeak           0.403951
ExerciseAngina    0.494282
HeartDisease      1.000000
Name: HeartDisease, dtype: float64
```

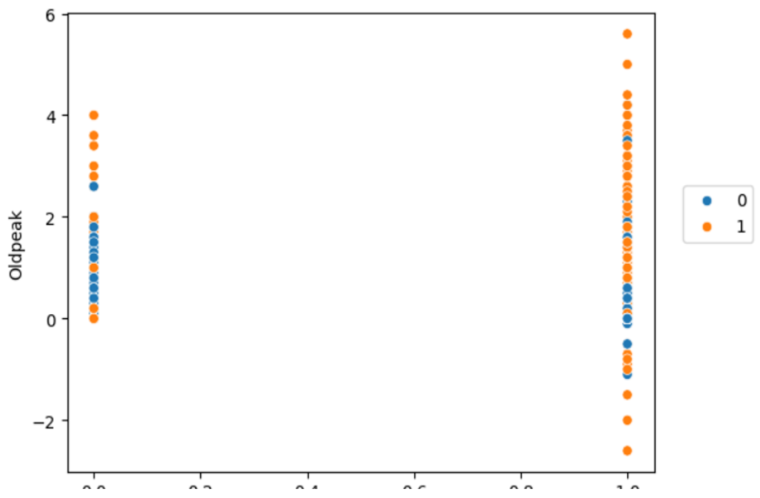
- STEP 4: The scatter plot which depicts after the removal Of the outliers

```
outlier=data[(data['Sex']==0) & (data['Oldpeak']>=6)]
index=outlier.index
```

```
data=data.drop(index)
data.reset_index(drop=True,inplace=True)
# removing the inconsistent data
```

```
sns.scatterplot(data=data,x='Sex',y='Oldpeak',hue='HeartDisease')
plt.legend(loc=(1.05,0.5))
```

```
<matplotlib.legend.Legend at 0x2be24bb1810>
```



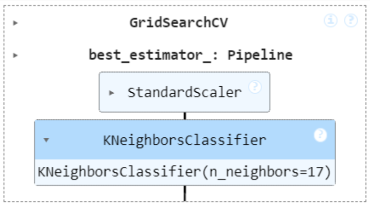
STEP 5: The knn model is created and the data is split using the train_test_split method of sklearn. A pipeline is created which performs the necessary operations such as the scaling and knn model creation. Grid search CV is used to find the best estimator for the model

```
58]: # standard scaler is used to form the co-efficients
     scaler=StandardScaler()
     X_train=scaler.fit_transform(X_train)
     X_test=scaler.transform(X_test)
```

```
60]: from sklearn.neighbors import KNeighborsClassifier
     knn=KNeighborsClassifier()
     knn
```

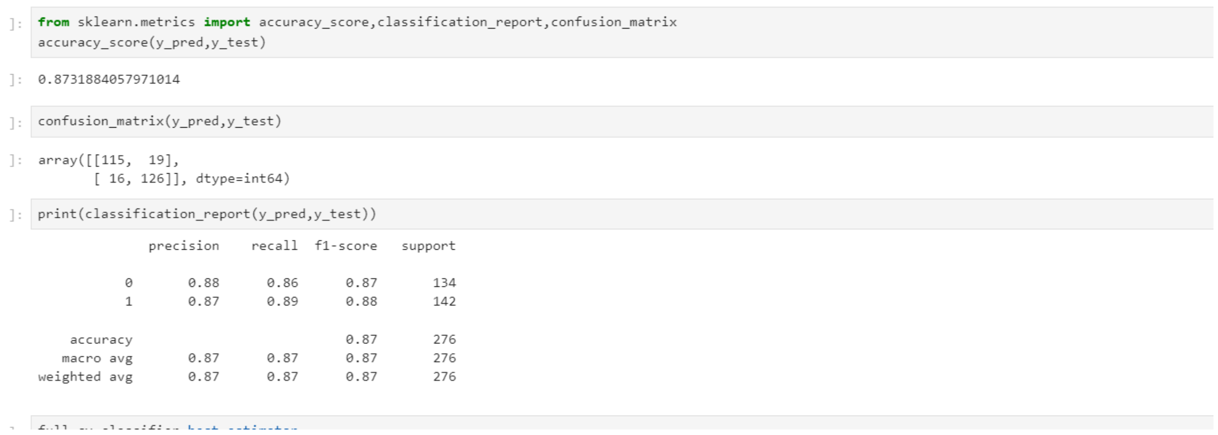
```
60]: KNeighborsClassifier
     KNeighborsClassifier()
```

```
61]: from sklearn.pipeline import Pipeline
     from sklearn.model_selection import GridSearchCV
     operations=[('scaler',scaler),('knn',knn)]
     pipe=Pipeline(operations)
     k_values=list(range(1,20))
     param_grid={'knn__n_neighbors':k_values}
     full_cv_classifier=GridSearchCV(pipe,param_grid,cv=5,scoring='accuracy')
     full_cv_classifier.fit(X_train,y_train)
```

61]:


```
GridSearchCV
best_estimator_: Pipeline
StandardScaler
KNeighborsClassifier
KNeighborsClassifier(n_neighbors=17)
```

- STEP 6: Different accuracy measuring methods such as the accuracy score, classification report and the confusion matrix is used



```
]: from sklearn.metrics import accuracy_score,classification_report,confusion_matrix
   accuracy_score(y_pred,y_test)

]: 0.8731884057971014

]: confusion_matrix(y_pred,y_test)

]: array([[115,  19],
          [ 16, 126]], dtype=int64)

]: print(classification_report(y_pred,y_test))

              precision    recall  f1-score   support

           0       0.88      0.86      0.87       134
           1       0.87      0.89      0.88       142

    accuracy                           0.87       276
   macro avg       0.87      0.87      0.87       276
weighted avg       0.87      0.87      0.87       276
```

GOAL : The accuracy of 87% accuracy rate for prediction , in which the best neighbor is 17.