EXPLORING DATASETS USING REGRESSION AND CLASSIFICATION MODELS FOR MACHINE LEARNING USING OPEN SOURCE DATA

SAMJEH EMMANUEL

Task 1: Regression models used for the Dataset 'HOUSE PRICE DATA FROM KING COUNTY'. This data represents the factors which influence housing prices in King County.

1.1 The first model used to train the dataset is Linear Regression.

There are screenshots on appendix 1 for the algorithm used.

1.2 The second model used is the Decision Tree Regressor.

There are screenshots on appendix 2 for the algorithm used.

http://localhost:8888/lab/tree/Untitled6.ipynb

Task 2: Classification models used for the Dataset 'Heart.csv' which presents data on the factors that may increase or decrease the chances of having a heart attack.

2.1 The first model used is the Logistic Regressor.

Screenshots of the algorithm are in appendix 3.

2.2 The second model is the Support Vector classifier.

There are screenshots on appendix 4 for the algorithm used.

http://localhost:8888/lab/tree/Untitled8.ipynb

3.1 Assessing the accuracy of the Regression models.

The dataset used for the regression models, there are 16 columns and 21613 instances. In choosing this data i was looking for an output that can be continuous and features that will coorelate with such an output in this case house prices. It is easier to find the line of best fit with data that has some correlation.

The Linear Regression(LR) model was less accurate in predicting house prices than the Decision Tree (DT) model. Using the same parameters of the test size (20%) and the random state of 0, the initial R-squared(r2) values for the LR were 0.66 moving down to as we manipulated the values up to a random state of 10 with a value of 0.65. On the other hand, the r2 values for the DT model with the same initial parameters was 0.70 reaching a maximum of 0.92 when we used a random state of 5. Using max depth did not improve any of the models significantly indicating that the DT model was the better choice for training for this dataset.

This indicates that the data fitted the regression better with the DT model than it did with the LR model ensuring better predictions from the model.

3.2 The 2 models obviously performed differently indicating underlying differences.

3.2.1 The Linear regression model is a type of supervised learning algorithm which is used to evaluate the relationship between variables in a dataset. It is a mathematical method used for predictive analysis. It establishes the relationship between an independent variable(s) or features usually on the x-axis of a plotted graph and dependent variable or outcome usually plotted on the y-axis of the graph. The independent variables can be used to predict changes in the dependent variable and unit changes in the x –axis affect the y-axis (Maulud & Abdulazees).

In this present case, multiple features are involved so it is a multilinear regression. Linear regression attempts to find the best-fitting line that minimizes the difference between the actual data points and the predicted values. Here, considering we are dealing with multiple feature a hyperplane is fitted which best represents all the points. The independent variables are used as input for the model which then which then works to produce predictions as output.

3.2.2 Decision Tree Regression model is equally a supervised learning algorithm which uses nodes which represent decisions or tests on attributes, branches representing the outcome of the decisions and leaf notes representing the final predictions. The goal here is to predict the value of a target variable based on several inputs. This algorithm works top to bottom as at every step going down to the final prediction it chooses a variable that best splits the set of items at that level. This continues down until it reaches a step where it can no longer split therefore concluding its prediction.

Because it is versatile and can be used in both regression and classification, it is one of the more popular models. It does not need feature scaling or normalization as a Linear regression algorithm will need making it ideal for datasets where some features do not have a linear relationship with the target.

3.2.3 The Decision tree algorithm worked better with this dataset because most of the features, about 9 out of 15 independent variables had no linear relationship with the target, which was price. This meant we needed a model that could capture the non- linear relationship between the features and the target variable. Also, I did not make any changes to the features, that is no normalization of the features was done in preprocessing leaving the models to work with the data as it was. This is better managed by a decision tree model as we saw it produce a better r2 score of 92% compared to the 66% produced by the LR model.

4.0

Before going to model building, it is important to preprocess the available data to ensure the features are properly scaled for machine learning and unwanted features which do not add to the training removed. In the dataset used, 6 features did have linear relationship with the target including bathrooms, sqft_living,sqft_above ,sqft_basement, Lat and sqft_ft living15. This was clearly visualized when we plotted the graph using pair plot.

The above features when used together excluding the nine which showed no linear relationship at all, we can have a better fitting line across the data which will in turn improve prediction for our chosen models. When the house was built or when it was renovated did not have any long-term effect on the price meanwhile the n number of bedrooms had an effect only for an initial period with no subsequent increases in price seen with the increase in the number of rooms as other factors in this county seem to be more important to buyers.

More data could be added about buyer income and age. Domain knowledge will play a big role as it will help in the search of the appropriate data needed to draw certain conclusions. And finally, we can always use even more models to see which one works best with our data.

Task 2 : In this task we used Classification models to train the ‘heart.csv’ dataset which had a binary target- more risk of having a heart attack or less risk of having heart attack.

2.1 The first model used is the Logistic Regression algorithm (LR).

Attached screenshots in appendix 3 show the source code and accuracy/recall results.

2.2 The second algorithm used is the Support Vector classification algorithm (SVM).

Attached screen shots in appendix 4 show the working and source code.

2.3 Assessing the accuracy of the above models.

The dataset used here is the ‘Heart Condition Data’ which has 14 features and 303 instances. For machine learning purposes the data has as target the output which is more likely to have a heart attack represented by the integer 1 or less likely to get a heart attack represented by 0.

For LR, testing after model training showed on the confusion matrix a precision for ‘0’ of 1.00, recall of 0.81 and 0.90 f1-score predicting a total of 26 out of 32 correctly. For the target ‘1’, precision was 0.83, recall 1.00 and f1-score 0.91 with all 29 predictions correct. Meanwhile the SVM model, for ‘0’ target, precision was 0.88, recall 0.44 and f1-score 0.58 with 14 out of 32 predictions correct. For the ‘1’ target, precision was 0.60, recall 0.93, and f1-score 0.73 with 27 out of 29 correct predictions correct.

The above shows generally that the LR performed better across the different metrics reaching a 100% recall for the ‘1’ target.

These results were achieved with a test size of 20% and a random state of 2. Tweaking the values of these parameters reduced both recall and precision values.

2.4 The 2 models function differently thus the different results.

2.4.1 Logistic regression is a supervised machine learning model which uses linear regression to find the line of best fit in a dataset. This establishes the relationship between the features and target variables. It is used mainly for modelling when we have dichotomous results like we have in the case of the data above, 1 or 0. It is different from Linear Regression in that the dependent variable rather than being continuous is categorical. So here, a statistical method where a logistic function is used to model the conditional probability (R.Khandelwal).

2.4.2 For the SVM classification model, there is focus on finding the maximum separating hyperplane between different classes of the target. It is equally a supervised machine learning algorithm and can be used both in regression models and in classification models. This model ensures that margins between the support vectors, which are the closest points of the different classes, are maximized. A hyperplane, which is the decision boundary is chosen whose distance from the nearest point from each of the classes is maximum. This hyperplane is used to separate the different classes.

2.4.3 The LR model worked better here mainly because the target for this dataset is binary. Given the overlapping of data points and thus classes, SVM will struggle to make sense of the noise, and this may affect the accuracy. SVM will perform best in situations where there is a need for optimal hyperplane separation of datapoints, for example in image classification where there is need for complete or maximum separation of classes.

2.4.4

Improving the models will entail preprocessing of the data to take out features which do not add to the model training especially for the LR which performs best when variables have some correlation. Domain knowledge in this case will see the possibility of adding features that will correlate better with the target and also under sampling to improve the linear relationship. Increasing the data to give more instances for the SVM to work with may improve its performance. Using sklearn.ensemble in python will allow the use of multiple SVM which will improve the general performance.

On the pair plot using seaborn, we that age, maximum heart rate achieved, cholesterol resting blood pressure and old peak have some relationship with the target. These features can be added to or the data improved by collecting over a longer period, so more instances are available for the model to work with. This alongside reducing dimensionality by taking out those features which do not affect the target variable. While attempting methods to improve the performance of the SVM, it remains vital to not overfit the data.

LR working.

jupyter Untitled6 Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[68]:
```
import seaborn as sb
import matplotlib as plt
import pandas as pd
import numpy as npy
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
from sklearn.tree import DecisionTreeRegressor
```

[69]:
```
dataset=pd.read_csv('kc_house_data.csv')
```

[70]:
```
dataset
```

[70]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	grade	sqft_above	sqft_basement	yr_built	yr_renovated	lat	long	sqft_living15	sqft_lot
0	221900.0	3	1.00	1180	5650	1.0	7	1180	0	1955	0	47.5112	-122.257	1340	56
1	538000.0	3	2.25	2570	7242	2.0	7	2170	400	1951	1991	47.7210	-122.319	1690	76
2	180000.0	2	1.00	770	10000	1.0	6	770	0	1933	0	47.7379	-122.233	2720	80
3	604000.0	4	3.00	1960	5000	1.0	7	1050	910	1965	0	47.5208	-122.393	1360	50
4	510000.0	3	2.00	1680	8080	1.0	8	1680	0	1987	0	47.6168	-122.045	1800	75
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	360000.0	3	2.50	1530	1131	3.0	8	1530	0	2009	0	47.6993	-122.346	1530	15
21609	400000.0	4	2.50	2310	5813	2.0	8	2310	0	2014	0	47.5107	-122.362	1830	72
21610	402101.0	2	0.75	1020	1350	2.0	7	1020	0	2009	0	47.5944	-122.299	1020	20
21611	400000.0	3	2.50	1600	2388	2.0	8	1600	0	2004	0	47.5345	-122.069	1410	12
21612	325000.0	2	0.75	1020	1076	2.0	7	1020	0	2008	0	47.5941	-122.299	1020	13

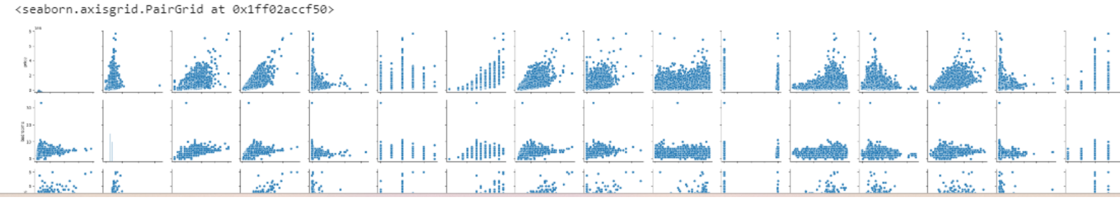
21613 rows × 16 columns

[71]:
```
'''using the housing data , i wll try to predict the price of houses in this area. So price will be my tartget.
I will first plot the data to see if the features correlate and if they are all useful for my regression model'''
```

[71]:
```
'using the housing data , i wll try to predict the price of houses in this area. So price will be my tartget.\nI will first plot the data to see if the f
eatures correlate and if they are all useful for my regression model'
```

[72]:
```
sb.pairplot(dataset)
```

[72]:
```
<seaborn.axisgrid.PairGrid at 0x1ff02accf50>
```



[115]:
```
'''from the graph there are seven features that have a positive correlation with the price, the others do not affect the price longterm or have no effect
on the price. I will train the model using all the initial data and then do same using data with only the 6 features that affect prices
across time'''
```

```
'from the graph there are seven features that have a positive correlation with the price, the others do not affect the price longterm or have no effect
\non the price. I will train the model using all the initial data and then do same using data with only the 6 features that affect prices \nacross time'
```

```
'''i will use Linear Regression as my first model'''
```

```
'i will use Linear Regression as my first model'
```

```
Y= dataset.iloc[:,0]
```

```
Y
```

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

```
X=dataset.iloc[:,1:]
```

```
X
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	grade	sqft_above	sqft_basement	yr_built	yr_renovated	lat	long	sqft_living15	sqft_lot15	condit

```
'from the graph there are seven features that have a positive correlation with the price, the others do not affect the price longterm or have no effect \non the price. I will train the model using all the initial data and then do same using data with only the 6 features that affect prices \nacross time'
```

```python
'''i will use Linear Regression as my first model'''
```

```
'i will use Linear Regression as my first model'
```

```python
Y= dataset.iloc[:,0]
```

```python
Y
```

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
```

```python
X=dataset.iloc[:,1:]
```

```python
X
```

	bedrooms	bathrooms	sqft_living	sqft_lot	floors	grade	sqft_above	sqft_basement	yr_built	yr_renovated	lat	long	sqft_living15	sqft_lot15	condit
0	3	1.00	1180	5650	1.0	7	1180	0	1955	0	47.5112	-122.257	1340	5650	
1	3	2.25	2570	7242	2.0	7	2170	400	1951	1991	47.7210	-122.319	1690	7639	
2	2	1.00	770	10000	1.0	6	770	0	1933	0	47.7379	-122.233	2720	8062	
3	4	3.00	1960	5000	1.0	7	1050	910	1965	0	47.5208	-122.393	1360	5000	
4	3	2.00	1680	8080	1.0	8	1680	0	1987	0	47.6168	-122.045	1800	7503	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
21608	3	2.50	1530	1131	3.0	8	1530	0	2009	0	47.6993	-122.346	1530	1509	
21609	4	2.50	2310	5813	2.0	8	2310	0	2014	0	47.5107	-122.362	1830	7200	
21610	2	0.75	1020	1350	2.0	7	1020	0	2009	0	47.5944	-122.299	1020	2007	
21611	3	2.50	1600	2388	2.0	8	1600	0	2004	0	47.5345	-122.069	1410	1287	
21612	2	0.75	1020	1076	2.0	7	1020	0	2008	0	47.5941	-122.299	1020	1357	

21613 rows × 15 columns

[80]:
```python
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.2,random_state=5)
```

[81]:
```python
regressor=LinearRegression()
regressor.fit(X_train,Y_train)
```

[81]: LinearRegression

[80]:
```python
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.2,random_state=5)
```

[81]:
```python
regressor=LinearRegression()
regressor.fit(X_train,Y_train)
```

[81]: LinearRegression


```
LinearRegression()
```

[82]:
```python
pred=regressor.predict(X_test)
print(Y)
print(pred)
```

```
0        221900.0
1        538000.0
2        180000.0
3        604000.0
4        510000.0
           ...
21608    360000.0
21609    400000.0
21610    402101.0
21611    400000.0
21612    325000.0
Name: price, Length: 21613, dtype: float64
[436861.95472713 148214.40321077 502261.38919104 ... 679576.42075764
 670858.41016788 490751.70013422]
```

[83]:
```python
r_square=metrics.r2_score(Y_test,pred)
print(r_square)
```

```python
r_square=metrics.r2_score(Y_test,pred)
print(r_square)
```

```
0.6596528488954259
```

```python
'''I will use the same file for the next model, decision tree regressor model'''
```

```
'I will use the same file for the next model, decision tree regressor model'
```

```python
'''analysis of the results will be developed in the report as different values for the random_state
were used producing different accuracy readings'''
```

```
'analysis of the results will be developed in the report as different values for the random_state\nwere used producing different accuracy readings'
```

```python
regressor=DecisionTreeRegressor(max_depth=50)
regressor.fit(X_train,Y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(max_depth=50)
```

```python
predictions=regressor.predict(X_test)
```

```python
print(predictions)
print(Y_test)
```

```
[355000. 245000. 417500. ... 590000. 565000. 578000.]
17485     365000.0
15164     225000.0
```

```python
X_train,X_test,Y_train,Y_test= train_test_split(X,Y,test_size=0.2,random_state=5)
```

```python
r_square= metrics.r2_score(Y_test,predictions)
print(r_square)
```

```
0.9294633238785864
```

The max_depth as captured on the screenshot was at a later tweaking of parameters. The r_score as seen above was what was had with no max_depth and with a test size of 0.2 and random state of 5.

jupyter Untitled8 Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[85]:
```python
import seaborn as sb
import numpy as np
import matplotlib as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn import svm
```

[2]:
```python
dataset=pd.read_csv('heart.csv')
```

[3]:
```python
dataset
```

[3]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

File Edit View Run Kernel Settings Help Trusted

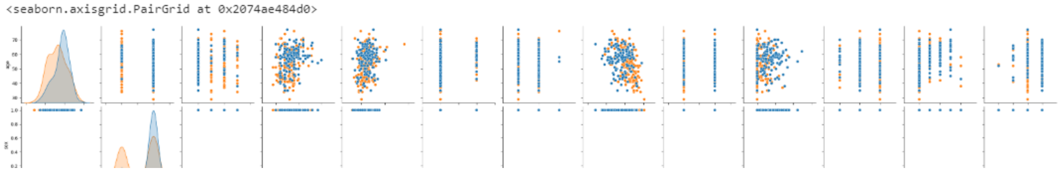
Code JupyterLab Python 3 (ipykernel)

[5]:
```python
'''To see the relationship the different features have with the tartget,'output',
I am going to plot them on a graph using pairplot from seaborn library '''
```

[5]: "To see the relationship the different features have with the tartget,'output', \nI am going to plot them on a graph using pairplot from seaborn library "

[22]:
```python
sb.pairplot(dataset,hue='output')
```

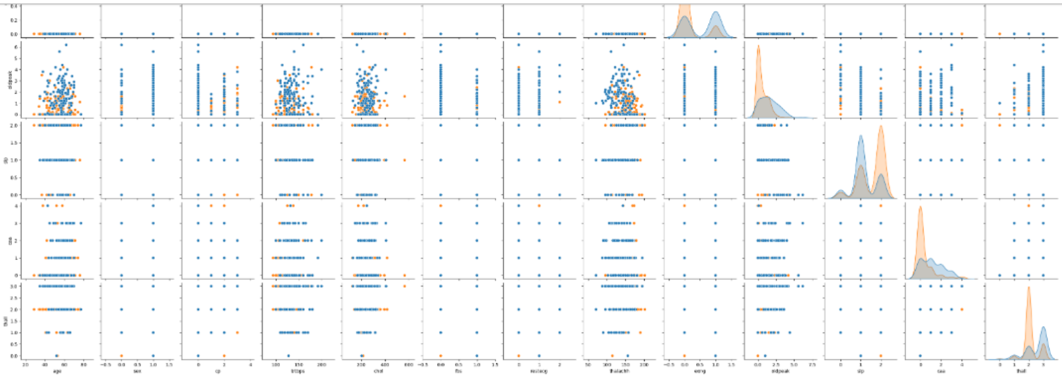
[22]: <seaborn.axisgrid.PairGrid at 0x2074ae484d0>



jupyter Untitled8 Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)



[7]:
```
''' the above graph does not show an explicit linear relationship between the featured and the output.
Since we are looking for one of 2 outcomes, we are going to use the Logistic Regression classifier
as our first classification model'''
```

[7]: ' the above graph does not show an explicit relationship between the featured and the output. \nSince we are looking for one of 2 outcomes, we are going to use the Logistic Regression classifier\nas our first classification model'

[10]:
```
Y=dataset.iloc[:,13]
```

jupyter Untitled8 Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[11]:
```
Y
```

[11]:
```
0      1
1      1
2      1
3      1
4      1
      ..
298    0
299    0
300    0
301    0
302    0
Name: output, Length: 303, dtype: int64
```

[12]:
```
X=dataset.iloc[:,0:12]
```

[13]:
```
X
```

[13]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa
0	63	1	3	145	233	1	0	150	0	2.3	0	0
1	37	1	2	130	250	0	1	187	0	3.5	0	0
2	41	0	1	130	204	0	0	172	0	1.4	2	0
3	56	1	1	120	236	0	1	178	0	0.8	2	0
4	57	0	0	120	354	0	1	163	1	0.6	2	0

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2
301	57	1	0	130	131	0	1	115	1	1.2	1	1
302	57	0	1	130	236	0	0	174	0	0.0	1	1

303 rows × 12 columns

[157]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.2,random_state=2)
```

[158]:
```
classifier=LogisticRegression()
classifier.fit(X_train,Y_train)
```

```
C:\Users\LORA\AppData\Local\Programs\Python\Python313\Lib\site-packages\sklearn\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

[158]: LogisticRegression

jupyter Untitled8 Last Checkpoint: yesterday

File Edit View Run Kernel Settings Help Trusted

Code JupyterLab Python 3 (ipykernel)

[158]: LogisticRegression

```
LogisticRegression()
```

[159]:
```
predictions=classifier.predict(X_test)
predictions
```

[159]:
```
array([1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1])
```

[160]:
```
Y_test
```

[160]:
```
99     1
296    0
89     1
30     1
234    0
      ..
173    0
94     1
161    1
216    0
91     1
Name: output, Length: 61, dtype: int64
```

[161]:
```
'''from the array above and comparing to the test, the predictions are not completely accurate. We will use a classification matrix to see exactly
how many predictions were correct or not using this model'''
```

```
[161]: 'from the array above and comparing to the test, the predictions are not completely accurate. We will use a classification matrix to see exactly \nhow ma
ny predictions were correct or not using this model'
```

```
[162]: print(classification_report(Y_test,predictions))
       print(confusion_matrix(Y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       1.00      0.81      0.90        32
           1       0.83      1.00      0.91        29

    accuracy                           0.90        61
   macro avg       0.91      0.91      0.90        61
weighted avg       0.92      0.90      0.90        61

[[26  6]
 [ 0 29]]
```

```
[163]: '''changing the values of the test size and random state gave us the best scores as seen above'''
[163]: 'changing the values of the test size and random state gave us the best scores as seen above'
```

```
[2]: '''The confusion matrix is a 2 by 2 matrix as our output is either 0 for less chance of heart attack or 1 for  more chance of heart attack'''
[2]: 'The confusion matrix is a 2 by 2 matrix as our output is either 0 for less chance of heart attack or 1 for  more chance of heart attack'
```

```
[165]: ''' The above matrix shows that our model classified 0 correctly 26 times out of a possible 32 and misclassified 6 times with f1-score 0.9 and recall of
[165]: ' The above matrix shows that our model classified 0 correctly 26 times out of a possible 32 and misclassified 6 times with f1-score 0.9 and recall of 0.
81'
```

```
[166]: "and it also predicted correctly the 29 out of 29 times in the test sample the 1's give an f1-score of 0.9 and recall of 0.91"
```

```
[167]: ''' Further analysis of the results in the report'''
[167]: ' Further analysis of the results in the report'
```

APPENDIX 4

```
[168]: '''I will now try another classification model for the same dataset, the support vector machine'''
[168]: 'I will now try another classification model for the same dataset, the support vector machine'
```

```
[169]: classifier=svm.SVC()
       classifier.fit(X_train,Y_train)
[169]: SVC
       SVC()
```

```
[170]: predictions=classifier.predict(X_test)
```

```
[171]: print(classification_report(Y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.88      0.44      0.58        32
           1       0.60      0.93      0.73        29

    accuracy                           0.67        61
   macro avg       0.74      0.68      0.66        61
weighted avg       0.74      0.67      0.65        61
```

```
[172]: print(confusion_matrix(Y_test,predictions))
```

```
[[14 18]
 [ 2 27]]
```

```
[172]: print(confusion_matrix(Y_test,predictions))

[[14 18]
 [ 2 27]]

[173]: ''' we see that for the exact same parameters as were used for the LogisticRegression model, that is test size of 20% and
random state of 2, we have a lower f1-score for both outputs 0.58 for 0 and 0.73 for 1 and the recall for the 0 output is even lower
 at 0.44, though recall for 1 is at 0.93'''

[173]: ' we see that for the exact same parameters as were used for the LogisticRegression model, that is test size of 20% and \nrandom state of 2, we have a lo
wer f1-score for both outputs 0.58 for 0 and 0.73 for 1 and the recall for the 0 output is even lower\n at 0.44, though recall for 1 is at 0.93'

[174]: '''Changing the parameters upwards or downwards only produced even worse predictions'''

[174]: 'Changing the parameters upwards or downwards only produced even worse predictions'

[175]: '''The rest of the comparisons will be found in the report'''

[175]: 'The rest of the comparisons will be found in the report'

[ ]:
```