Stats Final Project

Name: Kanishk Yadav

UTEP Id: 80772034

Project Topic: Breast Cancer Classification

Objective: During my machine learning project, our main objective was to classify the tumour as malignant and benign using different machine learning techniques.

Contents:-

Data importing and Exploration

>>> I imported the data using from the sklearn preloaded datasets and the imported data was stored in the variable "cancer".

The "cancer" was then described for exploring and I got the following output: -

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the
class

    :Attribute Information:
        - radius (mean of distances from center to points on the
perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of
the three
        worst/largest values) of these features were computed for
each image,
        resulting in 30 features.  For instance, field 0 is Mean
Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
    area (mean):                          143.5  2501.0
    smoothness (mean):                    0.053  0.163
    compactness (mean):                   0.019  0.345
    concavity (mean):                     0.0    0.427
    concave points (mean):                0.0    0.201
    symmetry (mean):                      0.106  0.304
    fractal dimension (mean):             0.05   0.097
    radius (standard error):              0.112  2.873
    texture (standard error):             0.36   4.885
    perimeter (standard error):           0.757  21.98
    area (standard error):                6.802  542.2
    smoothness (standard error):          0.002  0.031
    compactness (standard error):         0.002  0.135
    concavity (standard error):           0.0    0.396
    concave points (standard error):      0.0    0.053
    symmetry (standard error):            0.008  0.079
    fractal dimension (standard error):   0.001  0.03
    radius (worst):                       7.93   36.04
    texture (worst):                      12.02  49.54
    perimeter (worst):                    50.41  251.2
    area (worst):                         185.2  4254.0
    smoothness (worst):                   0.071  0.223
    compactness (worst):                  0.027  1.058
    concavity (worst):                    0.0    1.252
    concave points (worst):               0.0    0.291
    symmetry (worst):                     0.156  0.664
    fractal dimension (worst):            0.055  0.208
    ===================================== ====== ======
```

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree Construction Via Linear Programming." Proceedings of the 4th Midwest Artificial Intelligence and Cognitive Science Society, pp. 97-101, 1992], a classification method which uses linear programming to construct a decision tree. Relevant features were selected using an exhaustive search in the space of 1-4 features and 1-3 separating planes.

```
The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature
extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International
Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages
861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer
diagnosis and
     prognosis via linear programming. Operations Research, 43(4),
pages 570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine
learning techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer
Letters 77 (1994)
     163-171.
```

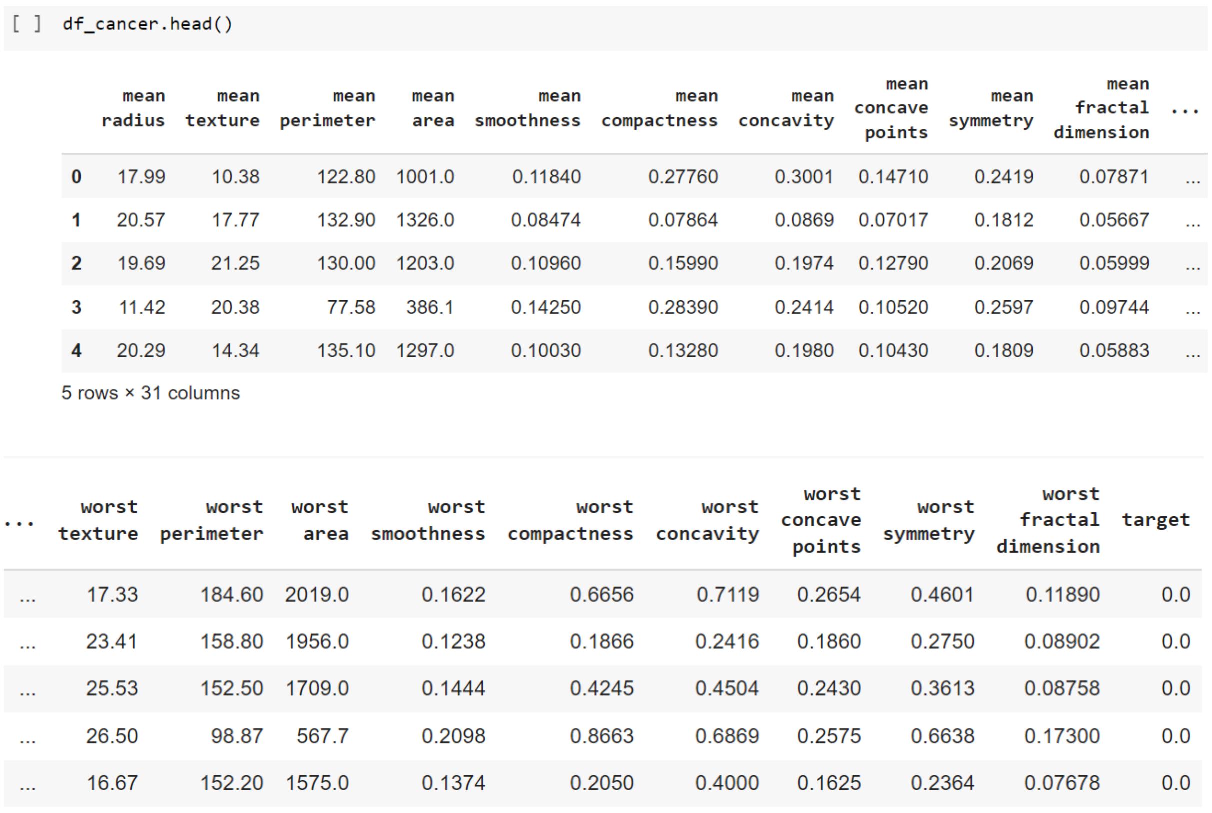
The above output tells me that there are 569 instances. Out of these 569, 212 are malignant and 357 are benign tumours. Since our data didn't have any missing attributed, we didn't need to replace or remove any NaN's or drop any rows.

There are in total 30 attributes, or we can say feature in the data. Some of them have been listed in the summary statistic table in the above output.

Other information such as References, etc., has also been provided.

>>> Other operations for exploration of data were further performed which were performed to see things such as "target names", "feature names" and seeing the shape of the data which was (569, 30)

>>> Further I converted the "cancer" data stored variable into a data frame "df_cancer". The ".head()" command for df_cancer gave following output



```
[ ] df_cancer.head()
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...

5 rows × 31 columns

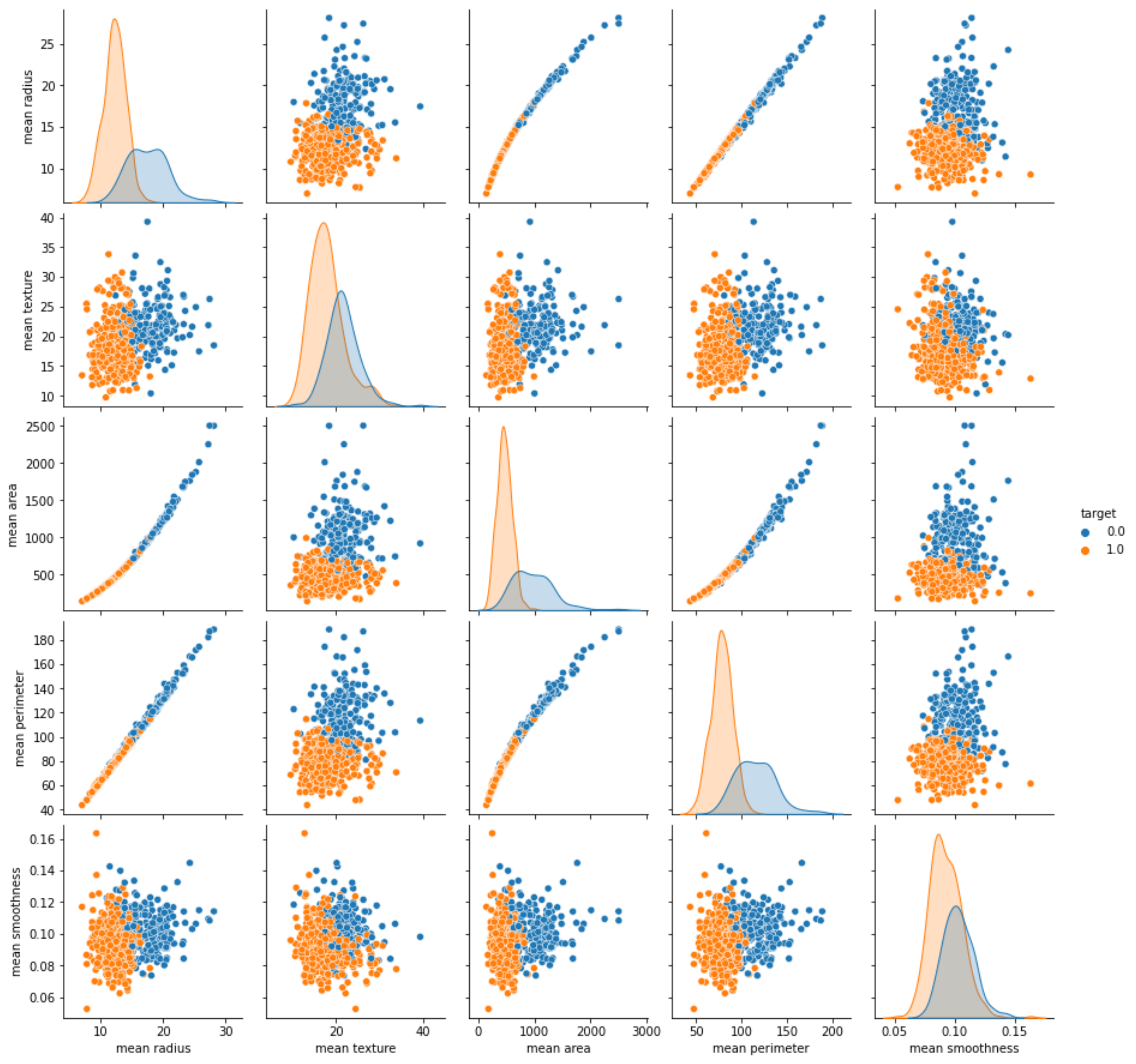
...	worst texture	worst perimeter	worst area	worst smoothness	worst compactness	worst concavity	worst concave points	worst symmetry	worst fractal dimension	target
...	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0.0
...	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0.0
...	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0.0
...	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0.0
...	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0.0

Here we see that the our "df_cancer" dataframe has dimension of 5 X 31 and also we observed that the "target" variable has 0's and 1's in the column (i.e., 0 for malignant and 1 for benign). By default .head() function gives only top 5 rows.

Data Visulization

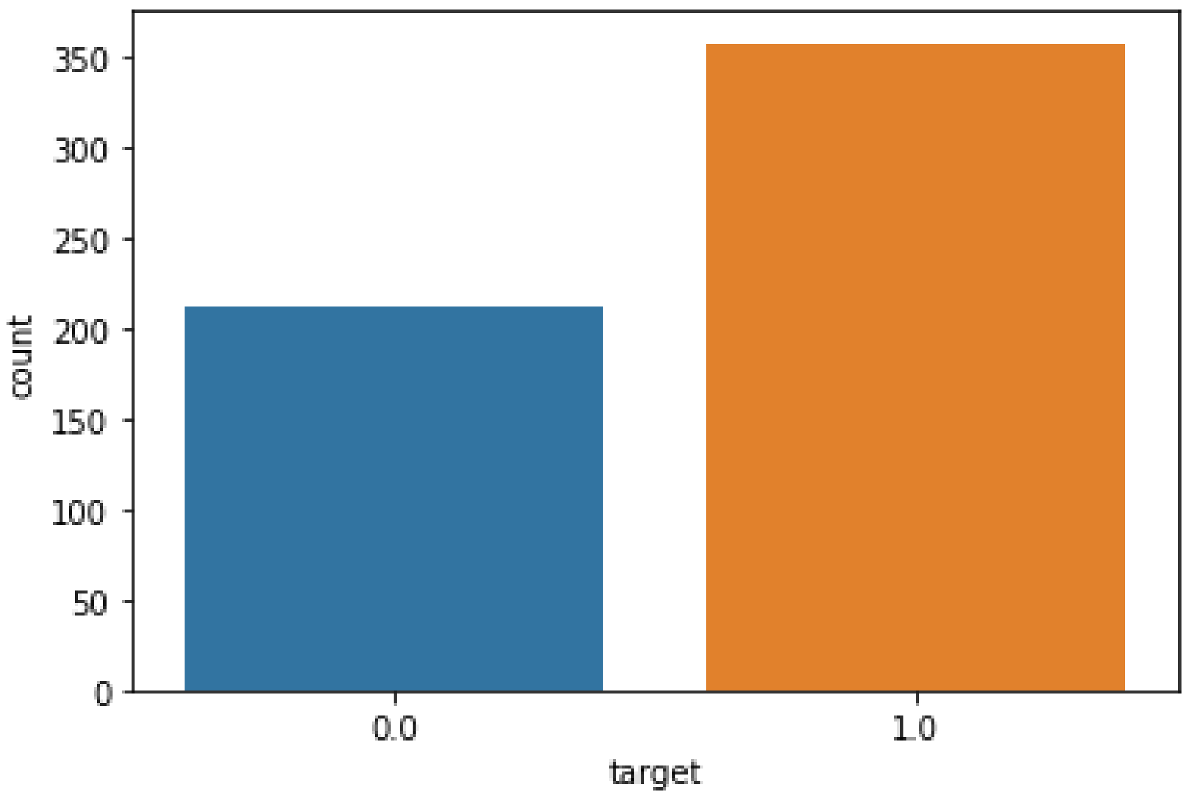
>>> During this portion of my project, I visualized the data to see how all the variables compare to each other and how machine learning is working. We used libraries seaborn and matplotlib.

We created a pairplot using seaborn to where hue was set to "target" variable. (Hue: Variable in data to map plot aspects to different colors and the variables were set to be: 'mean radius', 'mean texture', 'mean area', 'mean perimeter', 'mean smoothness'. The output for the pairplot is as follows:-



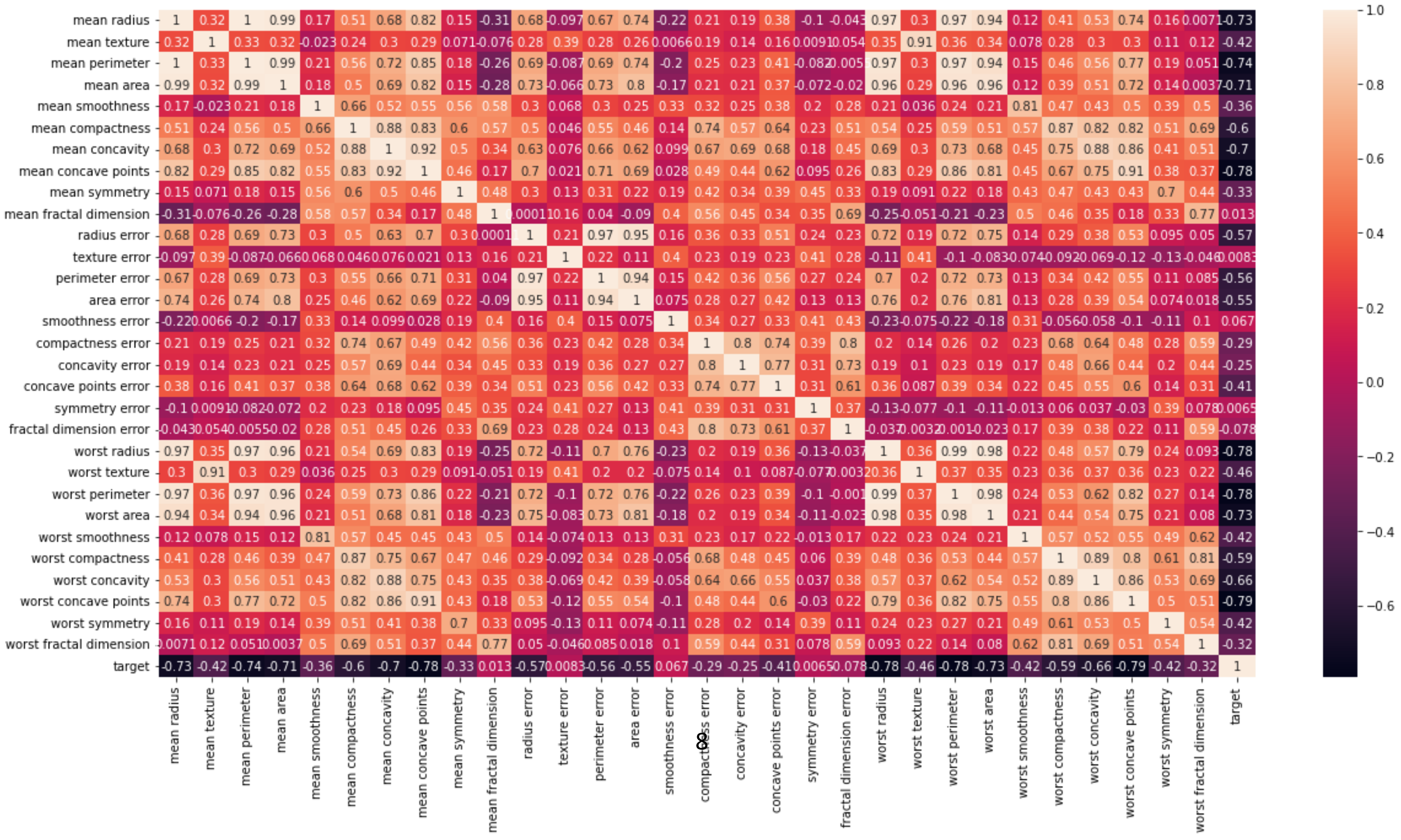
Here we can see the hue set by us “target”, gave the colours to our targets 0 (Malignant) as Blue and 1 (Benign) as Orange. We can see the interpretations here such as that malignant tumour seem to have higher mean perimeter and mean radius then as compared to benign, however, mean smoothness of benign variable is more that of malignant.

>>> We then drew the count plot to check the count of malignant and benign tumour. We got the following output for it:-



It is observable here that the benign tumour (orange) has higher count (which was also seen when we described that the data).

>>> We then drew a correlation matrix for df_cancer dataframe using seaborn library's heatmap, we got the following output:-

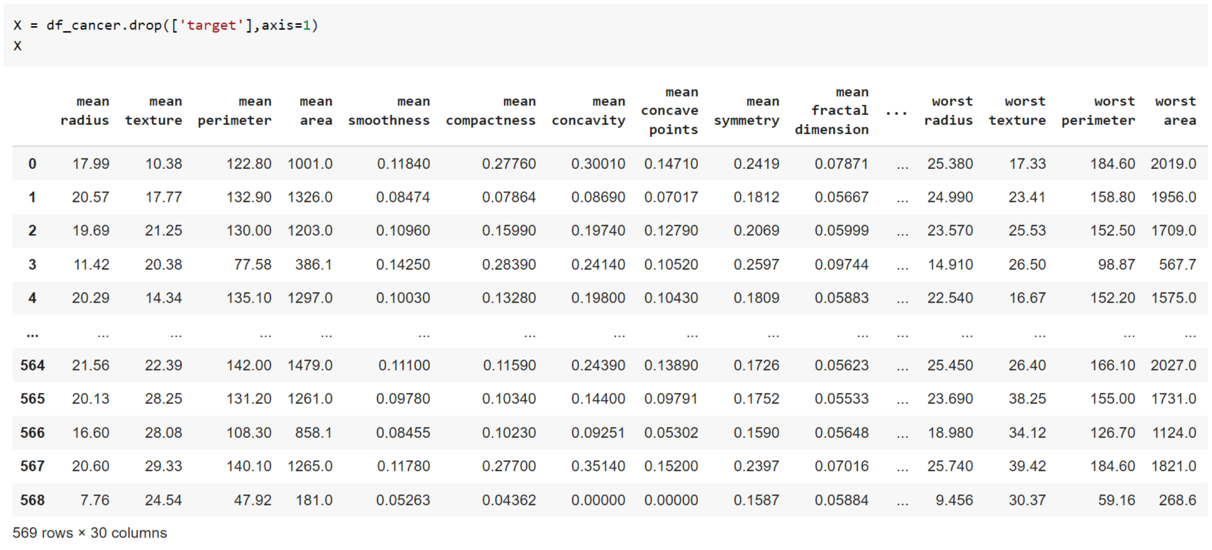


Here it can be observed that the heatmap is showing the relation between all the variables. We can see that lighter the colour of variable, the more correlation they have. The diagonal line in the centre with all the 1's depict the correlation among the same features.

Evaluating the model

>>> For evaluating the model, first we had to split the data into training and test set. Here we split the data as follows:-

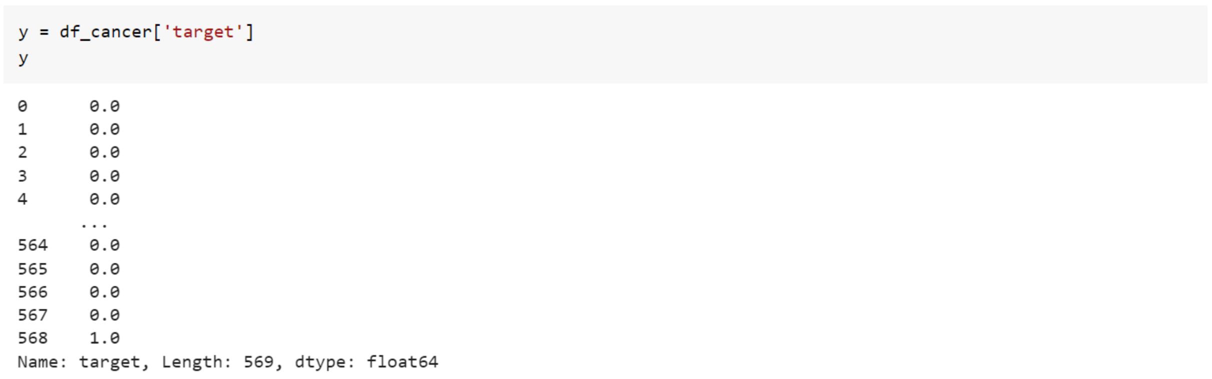
X is assigned as the part of the "df_cancer" such that it take all the features except the "target' and "y" variable takes all the target variable.



```
X = df_cancer.drop(['target'],axis=1)
X
```

	mean radius	mean texture	mean perimeter	mean area	mean smoothness	mean compactness	mean concavity	mean concave points	mean symmetry	mean fractal dimension	...	worst radius	worst texture	worst perimeter	worst area
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	0.2419	0.07871	...	25.380	17.33	184.60	2019.0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	0.1812	0.05667	...	24.990	23.41	158.80	1956.0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	0.2069	0.05999	...	23.570	25.53	152.50	1709.0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	0.2597	0.09744	...	14.910	26.50	98.87	567.7
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	0.1809	0.05883	...	22.540	16.67	152.20	1575.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
564	21.56	22.39	142.00	1479.0	0.11100	0.11590	0.24390	0.13890	0.1726	0.05623	...	25.450	26.40	166.10	2027.0
565	20.13	28.25	131.20	1261.0	0.09780	0.10340	0.14400	0.09791	0.1752	0.05533	...	23.690	38.25	155.00	1731.0
566	16.60	28.08	108.30	858.1	0.08455	0.10230	0.09251	0.05302	0.1590	0.05648	...	18.980	34.12	126.70	1124.0
567	20.60	29.33	140.10	1265.0	0.11780	0.27700	0.35140	0.15200	0.2397	0.07016	...	25.740	39.42	184.60	1821.0
568	7.76	24.54	47.92	181.0	0.05263	0.04362	0.00000	0.00000	0.1587	0.05884	...	9.456	30.37	59.16	268.6

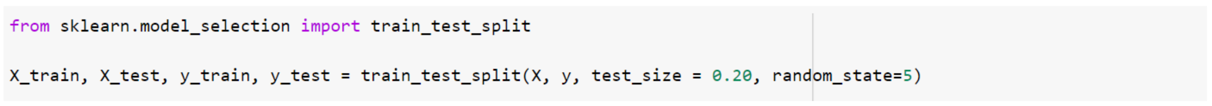
569 rows × 30 columns



```
y = df_cancer['target']
y
```

```
0      0.0
1      0.0
2      0.0
3      0.0
4      0.0
      ...
564    0.0
565    0.0
566    0.0
567    0.0
568    1.0
Name: target, Length: 569, dtype: float64
```

Using "train_test_split" from sklearn.model_selection,, we split the X and y as follows in the output:-



```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20, random_state=5)
```

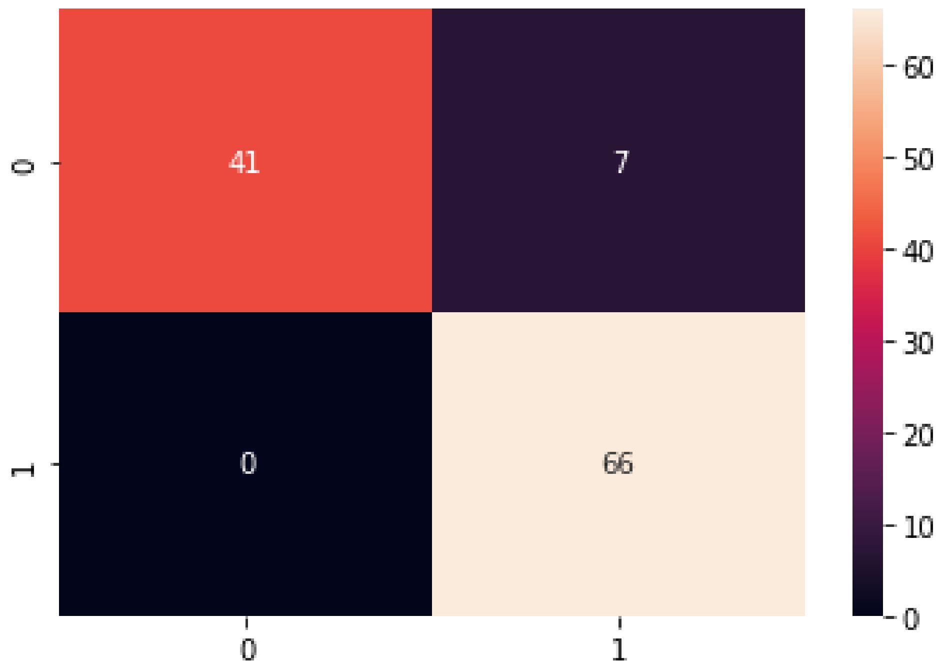
Model 1) Support Vector Machine:

Here we first stored the SVM algorithm into a variable and then applied to the X_test and y_test. Upon seeing the score of the SVC on X_test and y_test, we go the following output:-

```
svc_model.score(X_test, y_test)
```

```
0.9385964912280702
```

>>> Confusion matrix was then created to see the TP, TN, FP, FN. We created the confusion matrix using heatmap. Before that, we created a variable to store X_test called "y_predict_SVM". We got the following output:-



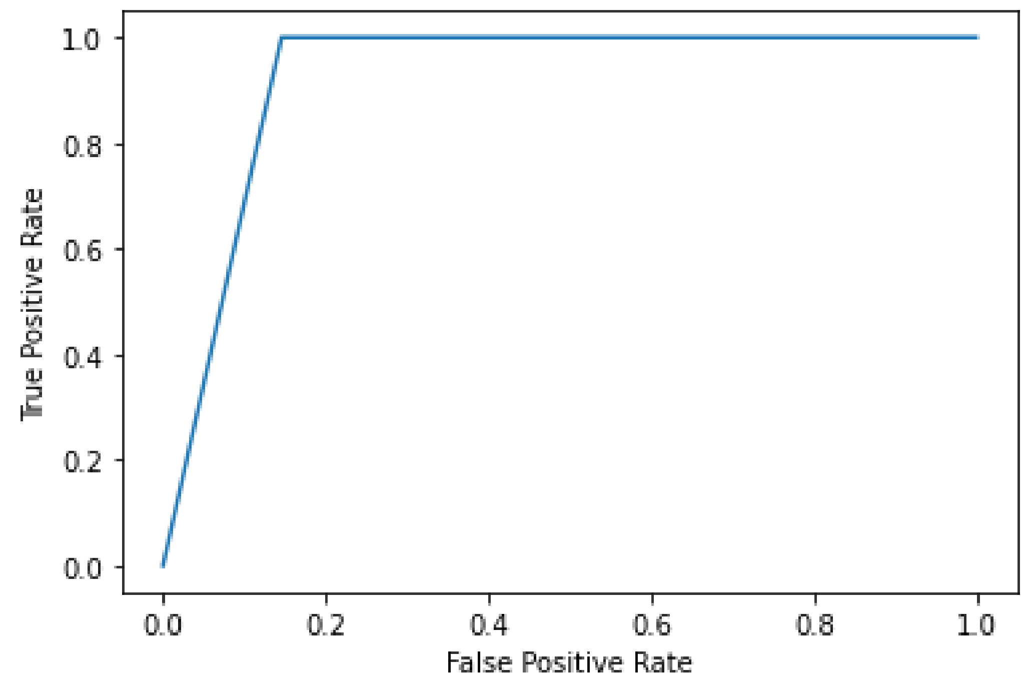
Here we observed the our model had very less amount of Type-I and Type-II error.

>>> We then further went on to make a classification report for the above model and got the following report for the classification report.

```
print(classification_report(y_test, y_predict_SVM))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.85      0.92        48
         1.0       0.90      1.00      0.95        66

    accuracy                           0.94       114
   macro avg       0.95      0.93      0.94       114
weighted avg       0.94      0.94      0.94       114
```

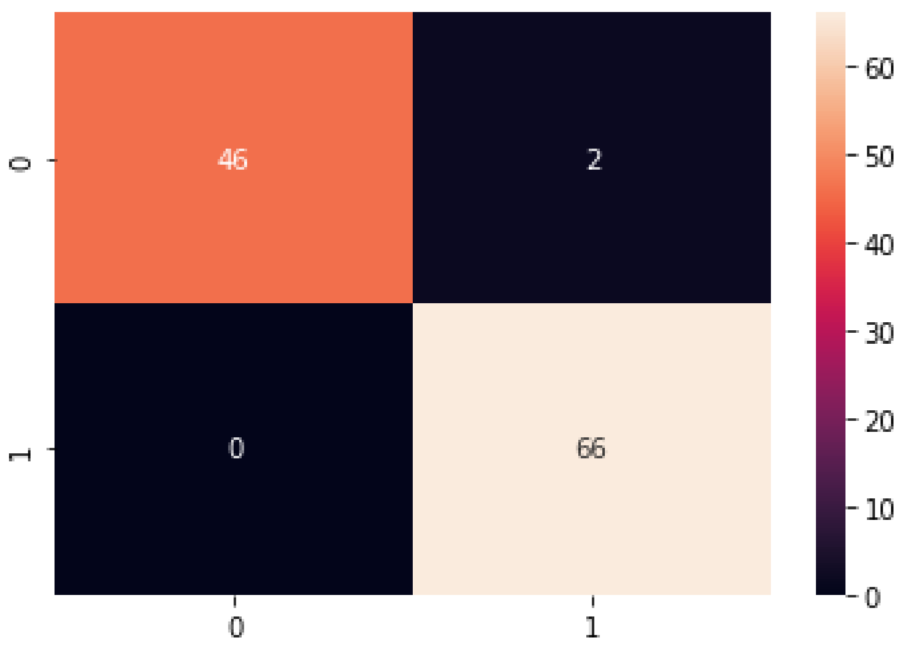
>>> ROC Curve was made for the SVM:



>>> Further the model for the SVM was improved by using normalization:-

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Now, after normalizing the data (or we can say doing min-max scaling), we further made confusion matrix and classification report and go the following output with the scales "X_train_scaled":-



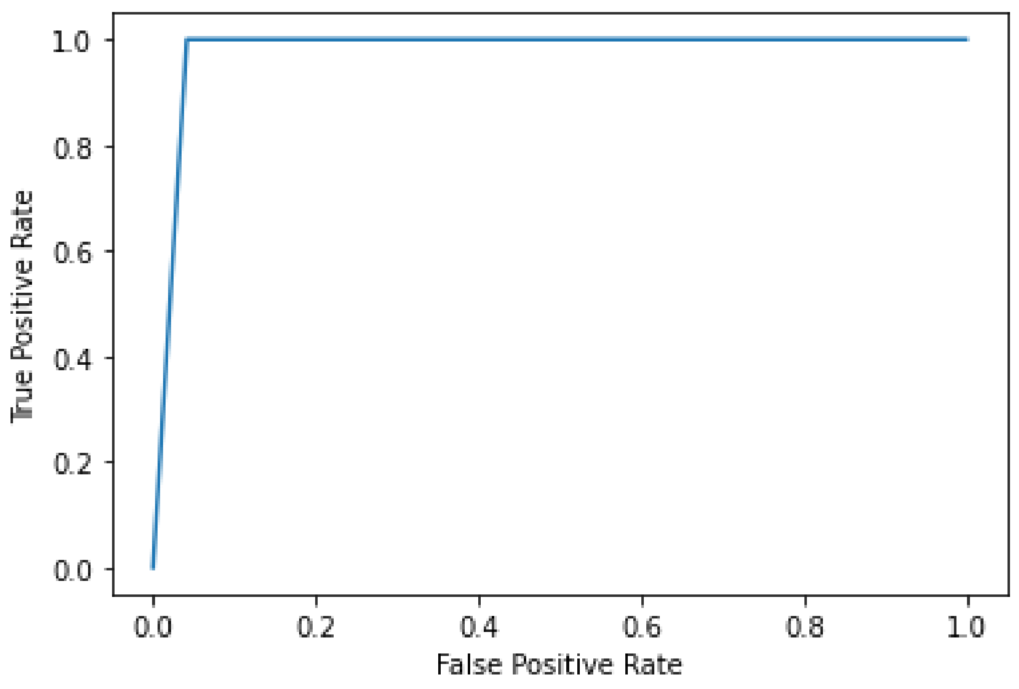
The heatmap above above shows slight reduction in Type-I error after model improvement of SVM by normalization and improvement in true positives.

Further the classification report after the normalization gave slight improvements too with increase in accuracy from 94 to 98.

```
print(classification_report(y_test,y_predict_svm_1))
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.96      0.98        48
         1.0       0.97      1.00      0.99        66

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

>>> ROC Curve was made for the SVM after Normalization:



Model 2) Random Forest:-

We then used an ensemble method called Random Forest.

Here again the, upon using the algorithm for random forest by storing it into a variable and applying it on the data, we got following score, confusion matrix and classification report. In our model we set the following parameters:

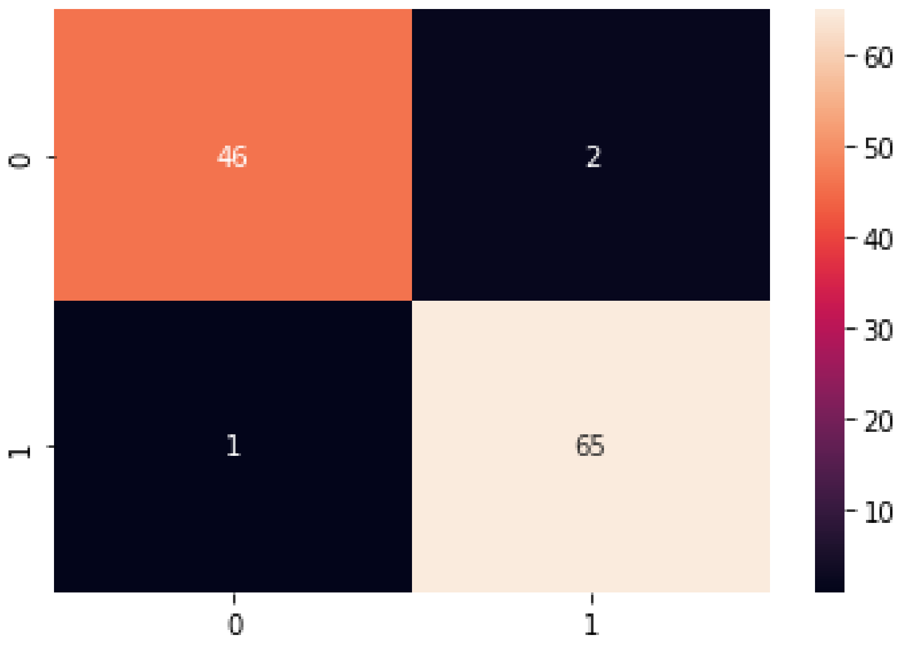
n_estimators = 100

n_jobs = -1 (meaning the CPU is instructed to use as many cores as available for use)

random_state = 100

```
random_forest.score(X_test, y_test)
```

```
0.9736842105263158
```



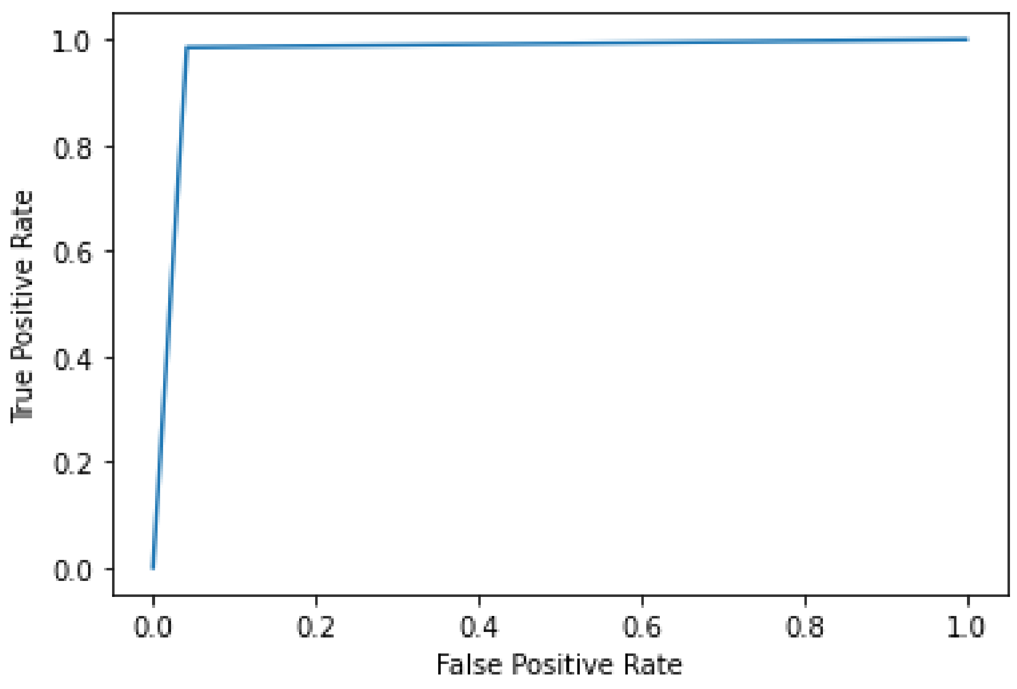
The confusion matrix clearly shows the Type-I and Type-II errors.

```
print(classification_report(y_test, y_pred_randomforest))
```

```
              precision    recall  f1-score   support

         0.0       0.98      0.96      0.97        48
         1.0       0.97      0.98      0.98        66

    accuracy                           0.97       114
   macro avg       0.97      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

We then got the classification report as above, we can see that the accuracy for it happens to be 97 which is very good.

>>> ROC Curve for the Random Forest:



Model 3) Logistic Regression

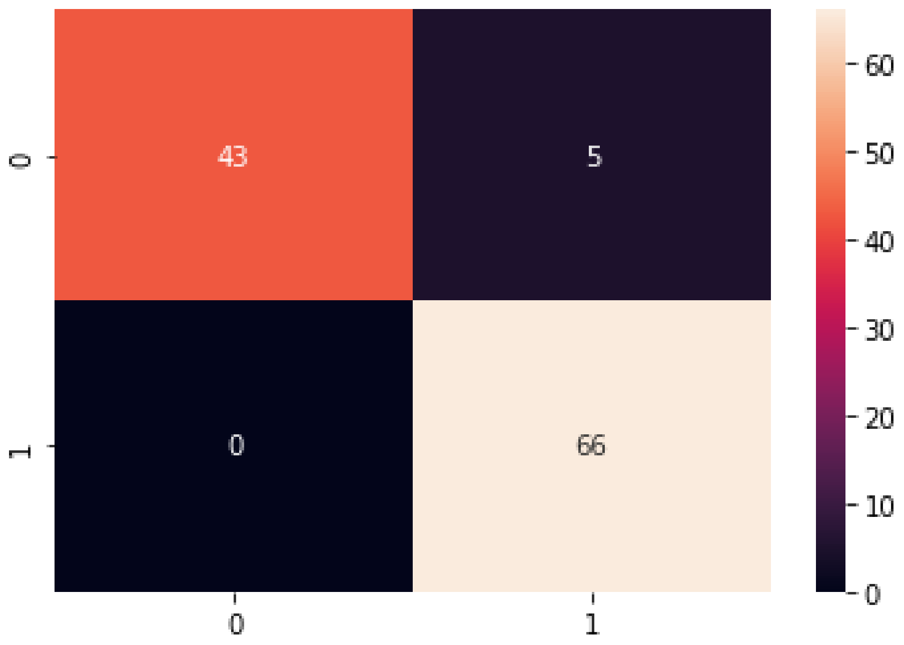
>>> We first imported the library for logistic regression and then stored the algorithm of logistic regression in a variable. We then fit the data (i.e., X_train, y_train) to the algorithm.

We got the following score for it:-

```
logmodel.score(X_test, y_test)
```

```
0.956140350877193
```

>>> Further we created the confusion matrix using the heat map to see the TP, TN, FP and FN. We observed that the Type – I and the Type – II errors were miniscule in count. True positives and False Negatives are very justifiable in it.

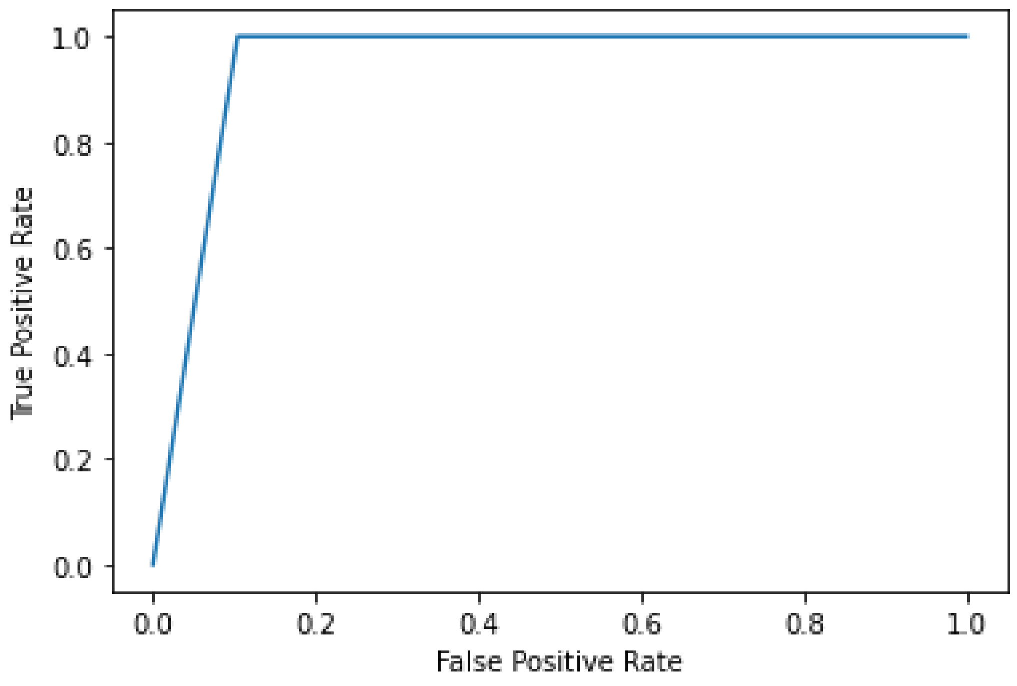


>>> We once again created classification matrix for the logistic regression to see accuracy of our model and we got the following scores:-

```
print(classification_report(y_test,y_pred_logistic_regression))
```

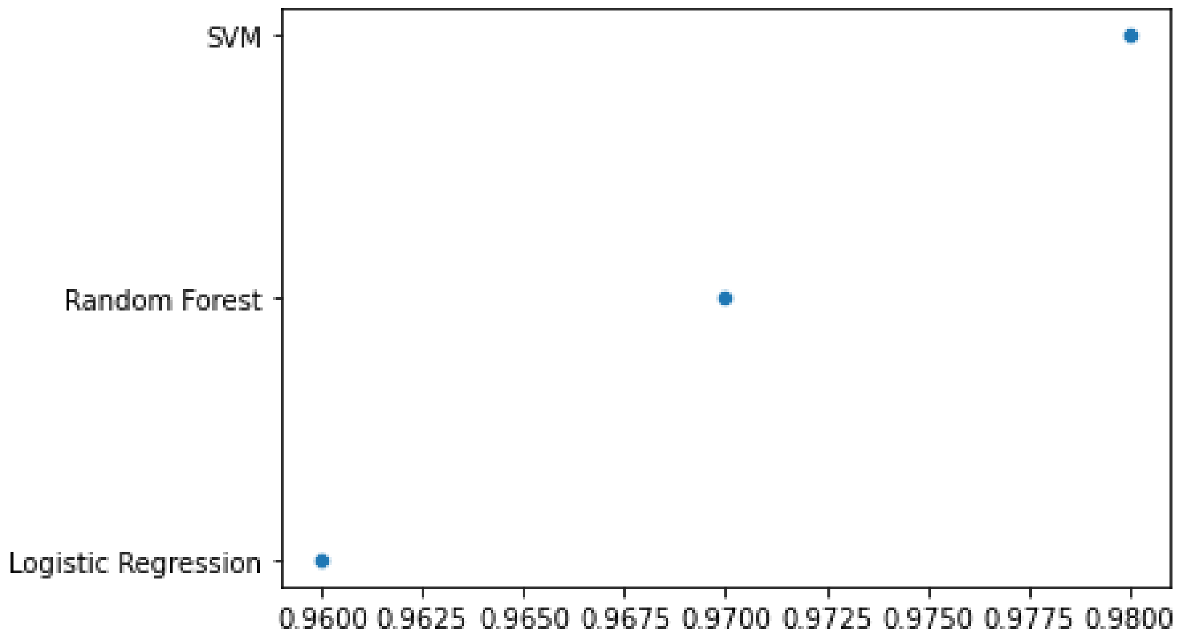
```
              precision    recall  f1-score   support

         0.0       1.00      0.90      0.95        48
         1.0       0.93      1.00      0.96        66

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

>>> ROC for the logistic regression:



Comparison of 3 models accuracies

After getting the accuracies for all the three models, we made a scatterplot of those for comparison of the 3 models we got:-



From the above scatterplot, it’s clearly observable that the accuracy of SVM are highest followed by Random Forest and Logistic Regression.

Conclusion

From analysing the data and applying all the algorithms, we were able to find that following was the order of the models to classify the given data correctly into benign and malignant tumours.

Support Vector Machine (with accuracy of 0.98 after normalization) > Random Forest (accuracy of 0.97) > Logistic regression (accuracy of 0.96)

We can conclude that, since all the models had >0.90 accuracy, it's clear that our model was able to classify the data to very good extent. Also, the heatmaps for the confusion matrix showed that there so less Type – I and Type –II errors in all three of our models suggesting less false positives and false negatives.

We can also conclude an obvious fact that SVM and Random Forest performed better as they are complex algorithms as compared to Logistic Regression which is much simpler as compared to SVM and Random Forest.

Appendix

Important codes:-

>>> Making Dataframe:

```
df_cancer = pd.DataFrame(np.c_[cancer['data'], cancer['target']], colum
ns = np.append(cancer['feature_names'], ['target']))
```

>>> Splitting data into training and testing:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0
.20, random_state=5)
```

>>> SVC, Random Forest and Logistic Regression importing and storing:

```
from sklearn.svm import SVC

from sklearn.metrics import classification_report, confusion_matrix

svc_model = SVC()



from sklearn.ensemble import RandomForestClassifier
```

```
random_forest = RandomForestClassifier(n_estimators=100, n_jobs = -
1, random_state = 100)



from sklearn.linear_model import LogisticRegression
logmodel = LogisticRegression()
```

>>> Normalization:

```
min_train = X_train.min()

min_train

range_train = (X_train - min_train).max()
range_train

X_train_scaled = (X_train - min_train)/range_train

X_train_scaled
```

>>> Correlation Matrix:-

```
plt.figure(figsize=(20,10))
sns.heatmap(df_cancer.corr(), annot=True)
```

>>> ACCURACY COMAPARISONS:-

```
import seaborn as sns
sns.scatterplot(x = (0.98, 0.97, 0.96), y = ("SVM", "Random Forest", "L
ogistic Regression"))
```