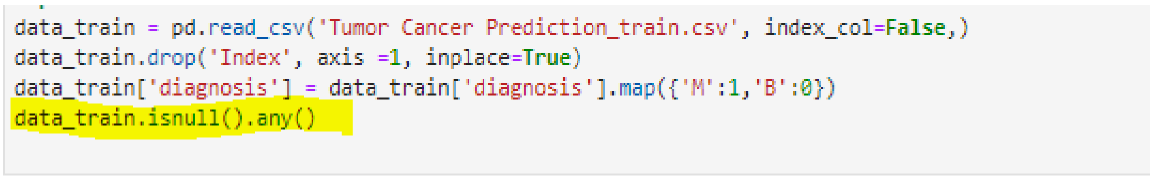
Project: Tumor Cancer Prediction.

Team members:

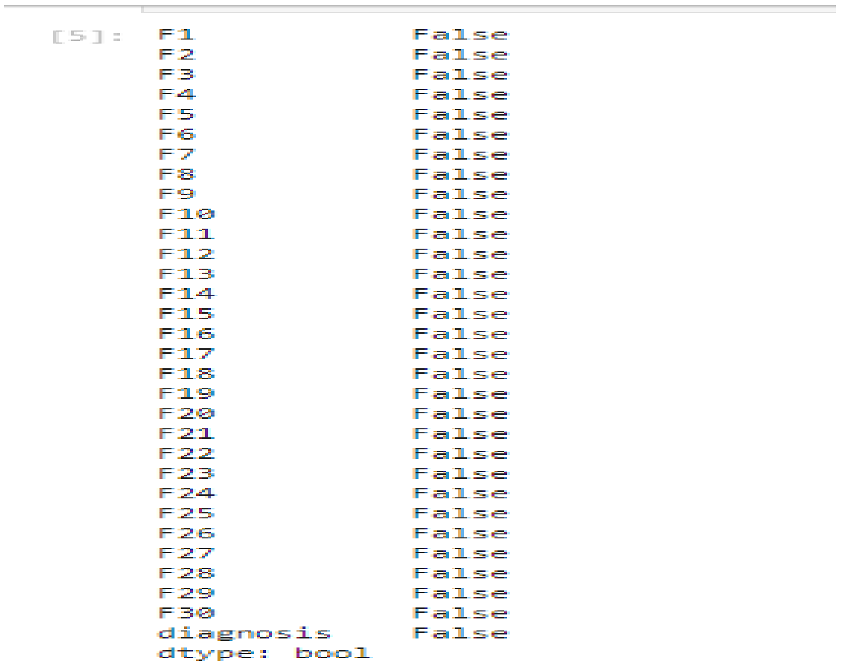
Rokaya abdelnaser ali	20161701028
Nada Mahmoud rezk	20161701069
Ola elwy abaza	20161701043
Salma ashraf abdelhalim	20161701033
Yomna hamed anter	20161701080

- preprocessing techniques.

- Check for missing value.



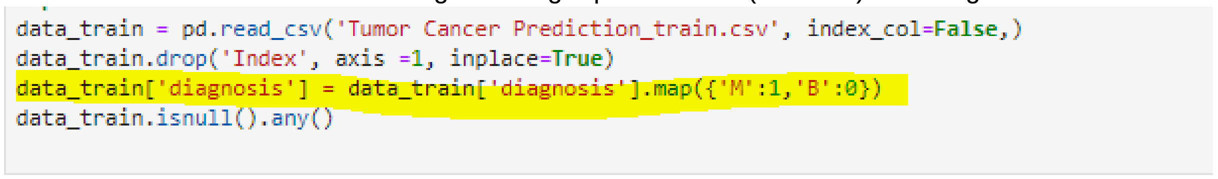
```
data_train = pd.read_csv('Tumor Cancer Prediction_train.csv', index_col=False,)
data_train.drop('Index', axis =1, inplace=True)
data_train['diagnosis'] = data_train['diagnosis'].map({'M':1,'B':0})
data_train.isnull().any()
```



```
[5]:  F1            False
      F2            False
      F3            False
      F4            False
      F5            False
      F6            False
      F7            False
      F8            False
      F9            False
      F10           False
      F11           False
      F12           False
      F13           False
      F14           False
      F15           False
      F16           False
      F17           False
      F18           False
      F19           False
      F20           False
      F21           False
      F22           False
      F23           False
      F24           False
      F25           False
      F26           False
      F27           False
      F28           False
      F29           False
      F30           False
      diagnosis     False
      dtype: bool
```

- map the class label

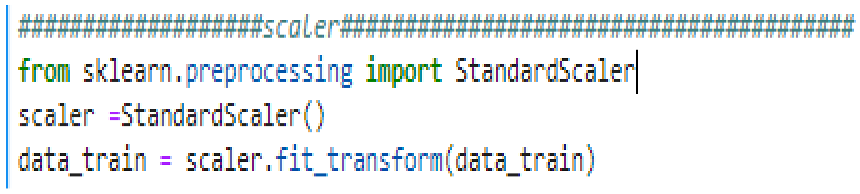
Transform the class labels from their original string representation (M and B) into integers



```
data_train = pd.read_csv('Tumor Cancer Prediction_train.csv', index_col=False,)
data_train.drop('Index', axis =1, inplace=True)
data_train['diagnosis'] = data_train['diagnosis'].map({'M':1,'B':0})
data_train.isnull().any()
```

- Feature Standardization.

  Use sklearn to scale and transform the data

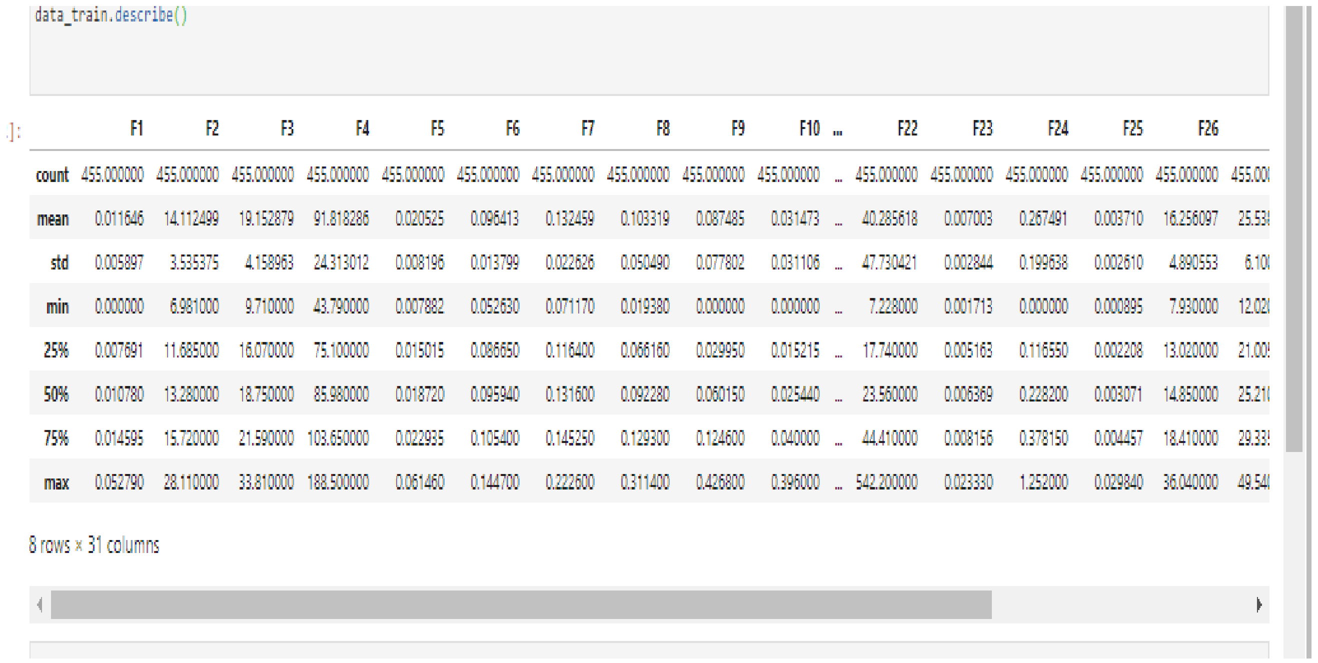


```
##################scaler######################################
from sklearn.preprocessing import StandardScaler
scaler =StandardScaler()
data_train = scaler.fit_transform(data_train)
```

-Data analysis.

- **DataFrame .describe()**

  Calculating some statistical data like **percentile, mean** and **std** of the numerical values of the Series or DataFrame.



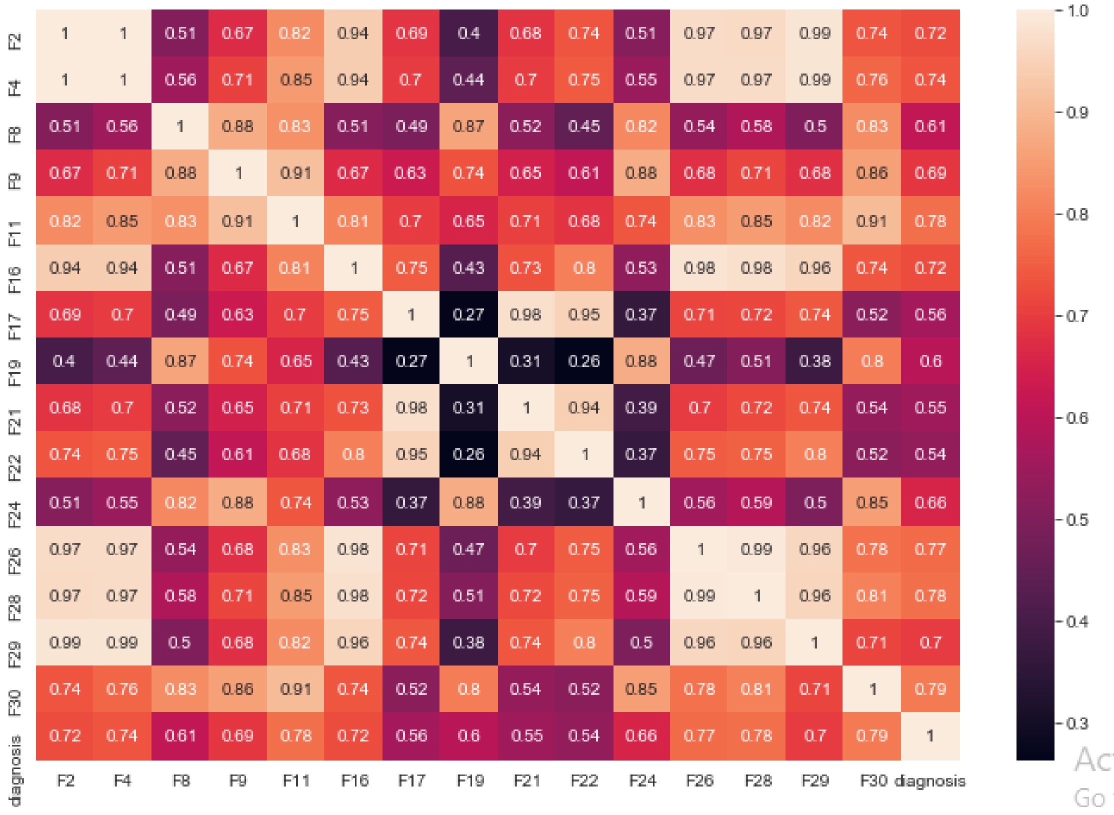
```
data_train.describe()
```

]:

	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	...	F22	F23	F24	F25	F26	
count	455.000000	455.000000	455.000000	455.000000	455.000000	455.000000	455.000000	455.000000	455.000000	455.000000	...	455.000000	455.000000	455.000000	455.000000	455.000000	455.00
mean	0.011646	14.112499	19.152879	91.818286	0.020525	0.096413	0.132459	0.103319	0.087485	0.031473	...	40.285618	0.007003	0.267491	0.003710	16.256097	25.53
std	0.005897	3.535375	4.158963	24.313012	0.008196	0.013799	0.022626	0.050490	0.077802	0.031106	...	47.730421	0.002844	0.199638	0.002610	4.890553	6.10
min	0.000000	6.981000	9.710000	43.790000	0.007882	0.052630	0.071170	0.019380	0.000000	0.000000	...	7.228000	0.001713	0.000000	0.000895	7.930000	12.02
25%	0.007691	11.685000	16.070000	75.100000	0.015015	0.086650	0.116400	0.066160	0.029950	0.015215	...	17.740000	0.005163	0.116550	0.002208	13.020000	21.00
50%	0.010780	13.280000	18.750000	85.980000	0.018720	0.095940	0.131600	0.092280	0.060150	0.025440	...	23.560000	0.006369	0.228200	0.003071	14.850000	25.21
75%	0.014595	15.720000	21.590000	103.650000	0.022935	0.105400	0.145250	0.129300	0.124600	0.040000	...	44.410000	0.008156	0.378150	0.004457	18.410000	29.33
max	0.052790	28.110000	33.810000	188.500000	0.061460	0.144700	0.222600	0.311400	0.426800	0.396000	...	542.200000	0.023330	1.252000	0.029840	36.040000	49.54

8 rows × 31 columns

- Correlation Matrix .



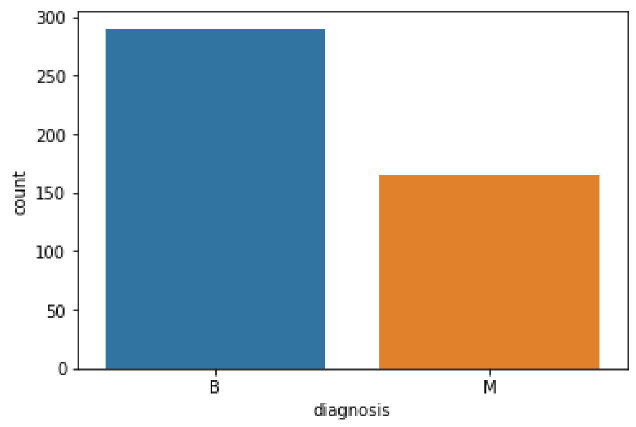
Observation:

- The f2 and f4 feature have a strong positive correlation with f6,f8 and f9 feature;
- The f21 and f22 feature have a weak correlation with f24,f8 and f19 feature;

- countplot

Observation:

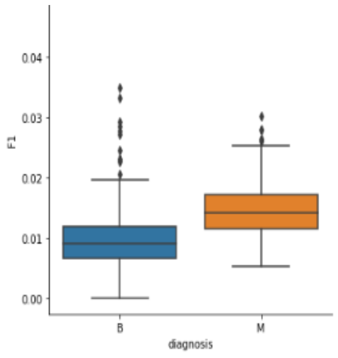
Number of benign tumor data more than number of malignant tumor data

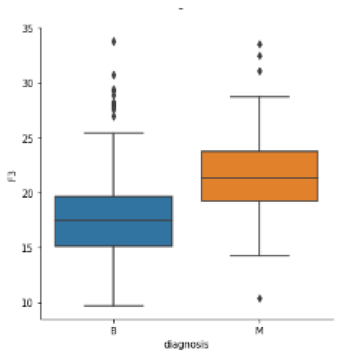


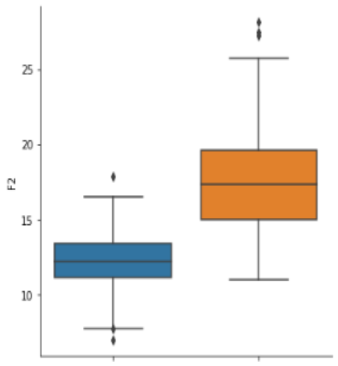
- boxplot

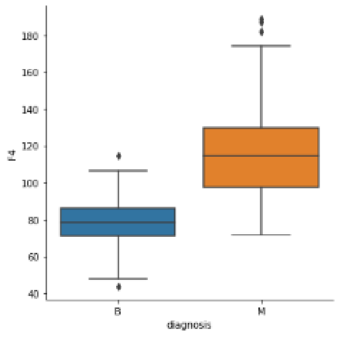
Observation:

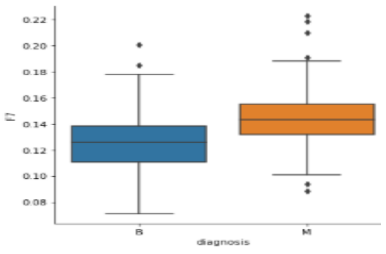
mostof the values are usually higher in malignant than that of benign

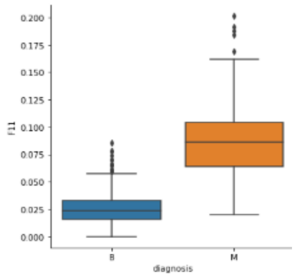


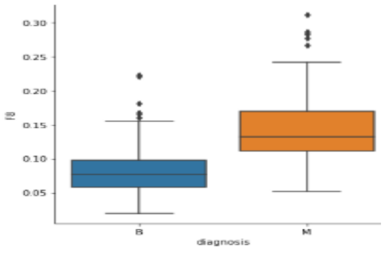


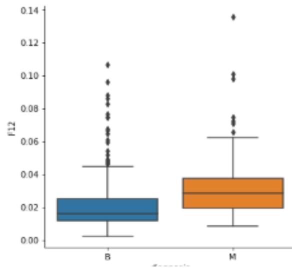






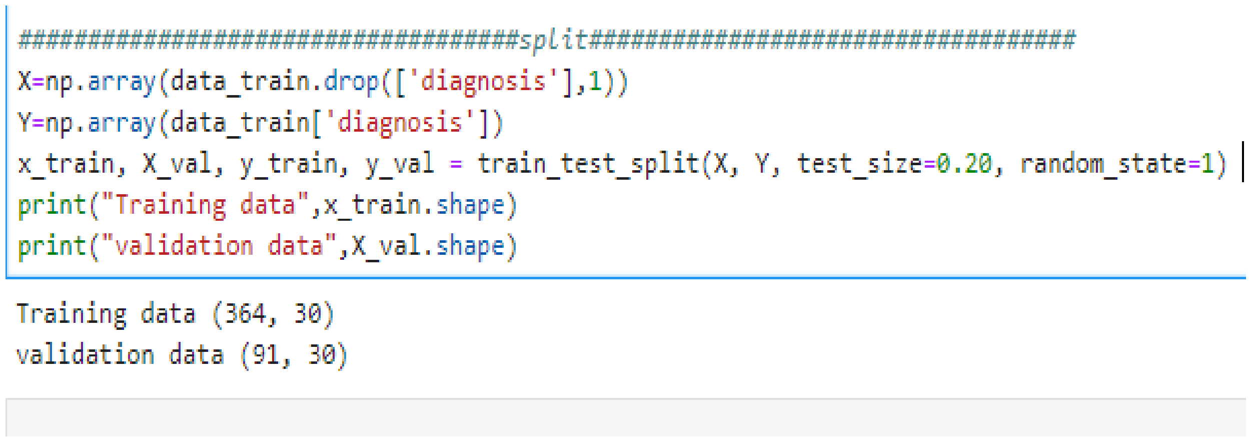






- Sizes of training and validation sets.

80% of the data for training and the remaining 20% for validation.



```
####################################split####################################
X=np.array(data_train.drop(['diagnosis'],1))
Y=np.array(data_train['diagnosis'])
x_train, X_val, y_train, y_val = train_test_split(X, Y, test_size=0.20, random_state=1)
print("Training data",x_train.shape)
print("validation data",X_val.shape)
```

```
Training data (364, 30)
validation data (91, 30)
```

- Hyperparameter tuning.

- **SVM**

  Hyperparameter:

  - Kernel:
    - sigmoid

      accuracy score = 0.945054945054945.
    - linear

      accuracy score = 0.978021978021978.
    - rbf

      accuracy score = 0.978021978021978.
  - gamma:
    - 0.001

      accuracy score = 0.9560439560439561.
    - 0.0001

      accuracy score = 0.7362637362637363.
    - 0.01

      Accuracy score = 0.978021978021978.

- **Decision Tree.**
  Hyperparameter:
  - max_depth:
    - (None)
      Accuracy score = 0.9340659340659341.
    - (2**)**
      Accuracy score = 0.9560439560439561.
    - (4**)**
      Accuracy score = 0.945054945054945
  - min_samples_leaf:
    - (10)
      accuracy score = 0.9560439560439561.
    - (6**)**
      Accuracy score = 0.967032967032967
    - (4**)**
      Accuracy score = 0.9340659340659341

- **xgboost.**
  Hyperparameter:
  - max_depth:
    - (3)
      accuracy score = 0.978021978021978
    - (2**)**
      Accuracy score = 0.967032967032967
    - (4**)**
      Accuracy score = 0.967032967032967

  - learning_rate:
    - (0.05)
      Accuracy score = 0.978021978021978.
    - (0.5**)**
      Accuracy score = 0.967032967032967
    - (0.10**)**
      Accuracy score = 0.978021978021978

- Dimensionality Reduction.

- **PCA**

  - SVM:
    - (0.90)
      Accuracy score = 0.978021978021978.
    - (0.50**)**
      Accuracy score = 0.9340659340659341
    - (25**)**
      Accuracy score = 0.978021978021978

.

  - Decision Tree:
    - (0.90)
      Accuracy score = 0.9340659340659341
    - (25**)**
      Accuracy score = 0.9340659340659341
    - (0.70**)**
      Accuracy score = 0.9230769230769231

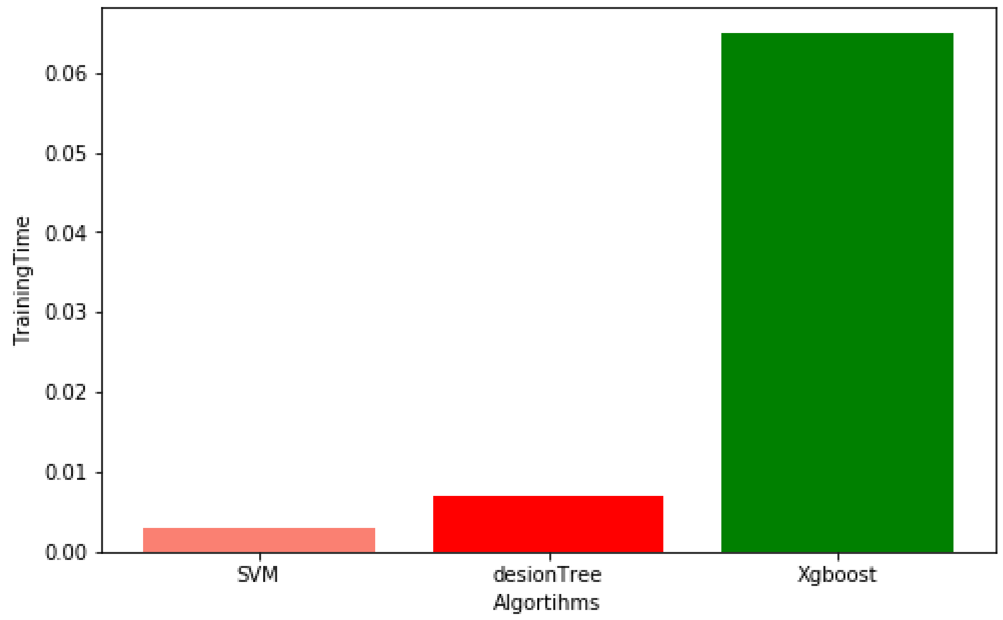
.

  - xgboost:
    - (0.90)
      Accuracy score = 0.945054945054945
    - (0.50**)**
      Accuracy score = 0.9340659340659341
    - (24**)**
      Accuracy score = 0.9560439560439561

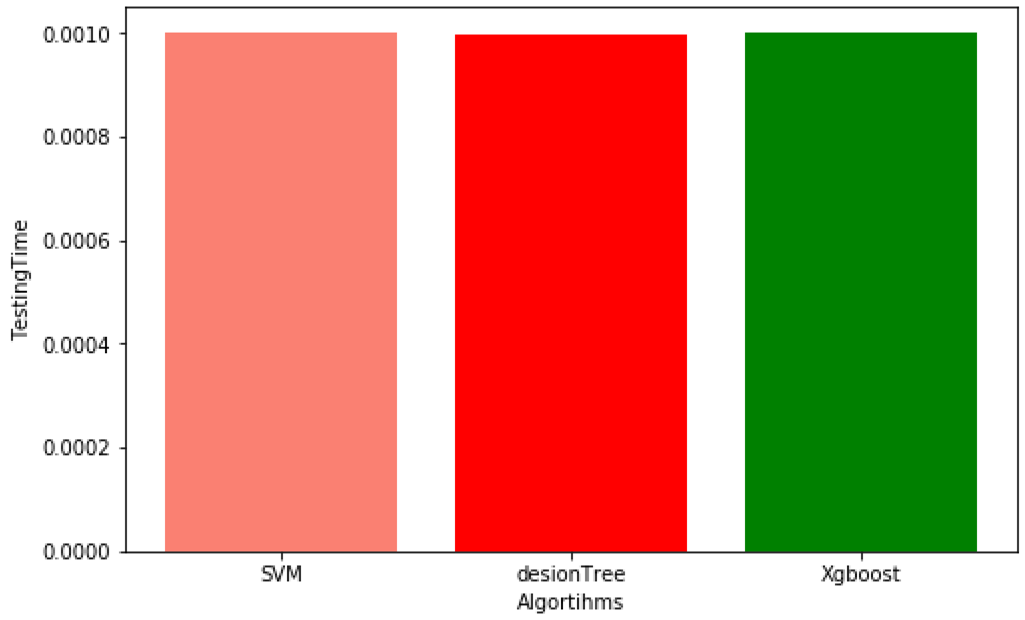
.

.

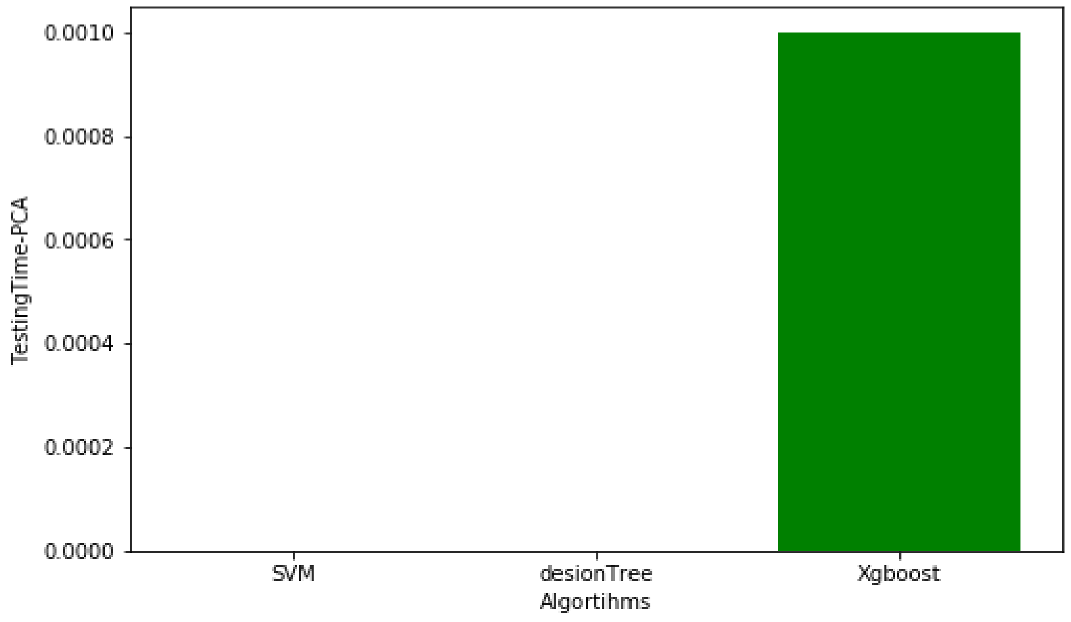
- **Training Time graph.**



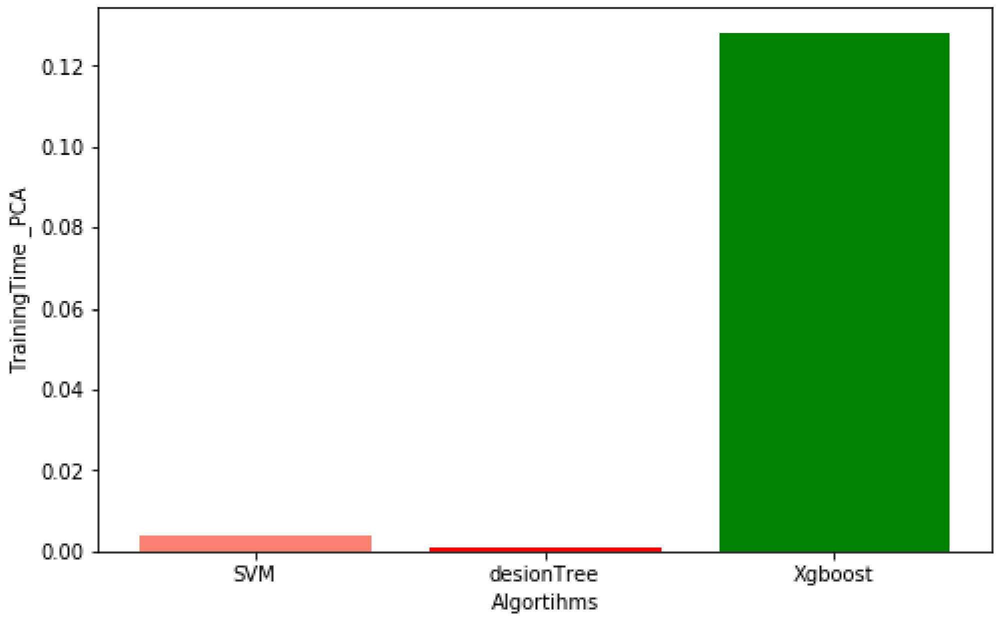
- **Testing Time graph.**



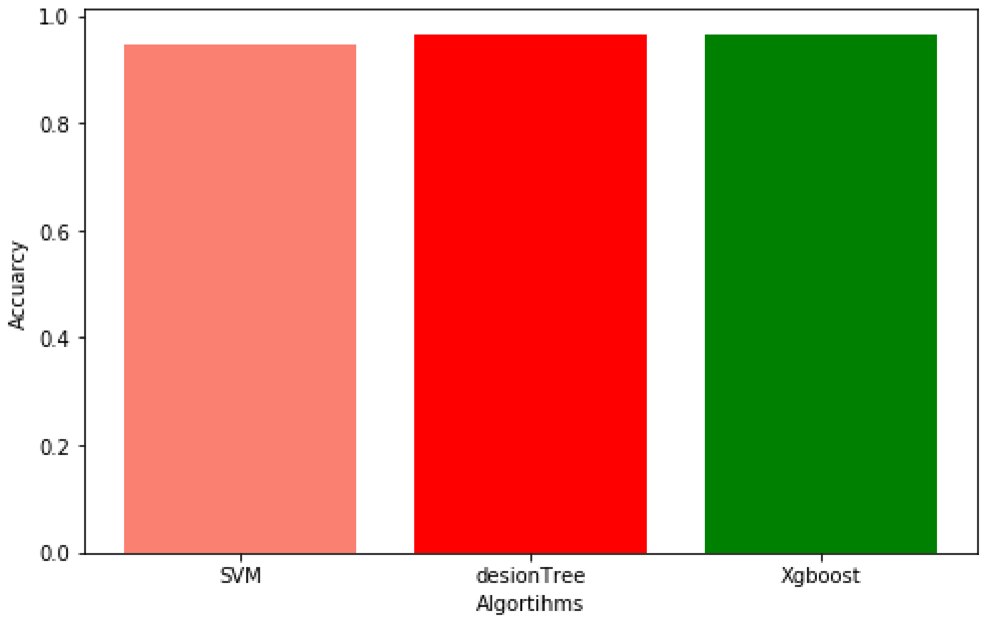
- **Testing Time graph with using PCA.**



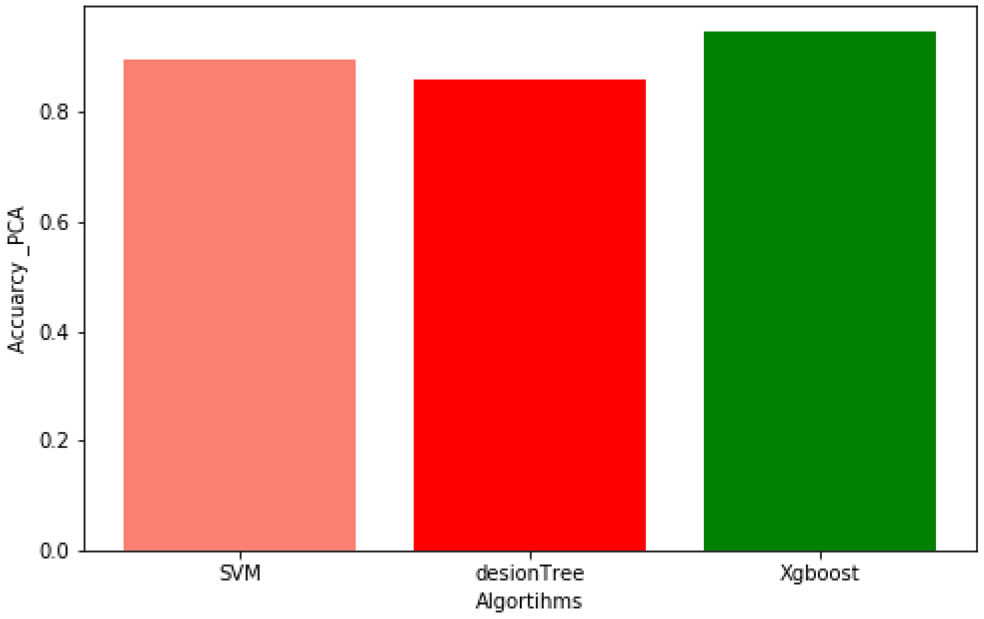
- **Training Time graph with using PCA.**



- **Accuracy graph**



- **Accuracy graph with using PCA**



Summary

We applied Decision Tree, XGBoosts and Support Vector Machine (SVM) algorithms to the Tumor Cancer dataset.
• To predict whether the Tumor cancer is malignant or benign.
• Compared the performance results of all the algorithms based on the accuracy values. and showed that XGBoosts classifier is the best among all in determining benign and malignant tumors.