Screenshots:

Initial data:

```
In [4]: df = pd.read_csv("data.csv")
        df.head(10)
```

Out[4]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.14710	...
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.07017	...
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.12790	...
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.10520	...
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.10430	...
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.08089	...
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.07400	...
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.05985	...
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.09353	...
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.08543	...

10 rows × 33 columns

Split the data into training data and split data.

```
[13]: from sklearn.model_selection import train_test_split

[14]: x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)

[15]: x_train.shape[1]

[15]: 30
```

Caption

Screenshots of running the Logistic Regression algorith

In [16]:
```
from sklearn.linear_model import LogisticRegression
lr = LogisticRegression()
lr.fit(x_train,y_train)
y_pred_lr = lr.predict(x_test)

import sklearn.metrics as metrics
fpr, tpr, threshold = metrics.roc_curve(y_test, y_pred_lr)
roc_auc = metrics.auc(fpr,tpr)
roc_auc
```

```
/Users/mounikasika/opt/anaconda3/lib/python3.9/site-packages/sklearn/linear_model/_logistic.py:814: ConvergenceWarn
ing: 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(
```

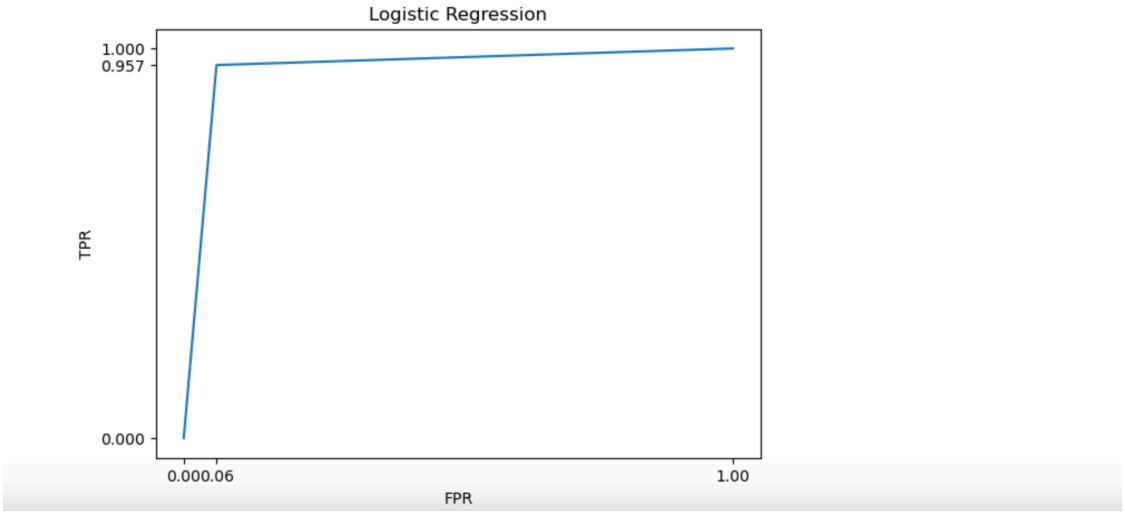
Out[16]:
```
0.9488726579866625
```

In [17]:
```
accuracy_score(y_test,y_pred_lr)
```

Out[17]:
```
0.9473684210526315
```

[18]:
```
plt.subplots(1, figsize=(7,5))
plt.title('Logistic Regression')
plt.yticks(tpr)
plt.xticks(fpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.plot(fpr, tpr)
```

t[18]:
```
[<matplotlib.lines.Line2D at 0x7f9829115b80>]
```



Caption

Running the KNN Algorithm:

In [19]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=3)
knn.fit(x_train,y_train)
y_pred_knn = knn.predict(x_test)
import sklearn.metrics
fpr, tpr, threshold = metrics.roc_curve(y_test,y_pred_knn)
roc_auc_knn = metrics.auc(fpr,tpr)
roc_auc_knn
```

```
/Users/mounikasika/opt/anaconda3/lib/python3.9/site-packages/sklearn/neighbors/_classification.py:228: FutureWarnin
g: Unlike other reduction functions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves t
he axis it acts along. In SciPy 1.11.0, this behavior will change: the default value of `keepdims` will become Fals
e, the `axis` over which the statistic is taken will be eliminated, and the value None will no longer be accepted.
Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

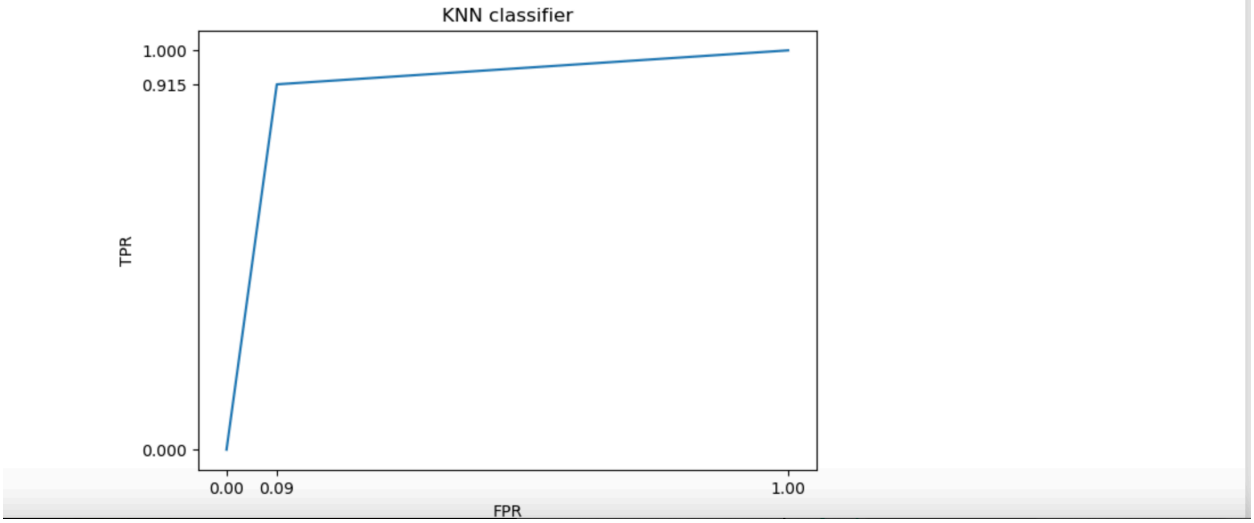
Out[19]:
```
0.9126706891076531
```

In [20]:
```
accuracy_score(y_test,y_pred_knn)
```

Out[20]:
```
0.9122807017543859
```

In [21]:
```
plt.subplots(1, figsize=(7,5))
plt.title('KNN classifier')
#plt.xticks(fpr)
plt.yticks(tpr)
plt.xticks(fpr)
plt.xlabel('FPR')
plt.ylabel('TPR')
plt.plot(fpr, tpr)
```

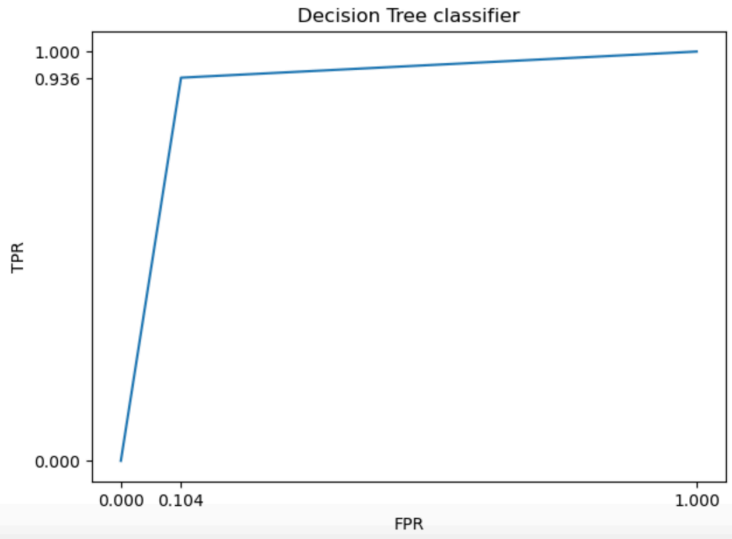
Out[21]:
```
[<matplotlib.lines.Line2D at 0x7f97f814b760>]
```



Running the Decision Tree algorithm.

```
In [22]: from sklearn.tree import DecisionTreeClassifier
         dt = DecisionTreeClassifier()
         dt.fit(x_train,y_train)
         y_pred_dt = dt.predict(x_test)
         import sklearn.metrics
         fpr, tpr, threshold = metrics.roc_curve(y_test,y_pred_dt)
         roc_auc_dt = metrics.auc(fpr,tpr)
         roc_auc_dt

Out[22]: 0.9158463004128296

In [23]: accuracy_score(y_test,y_pred_dt)

Out[23]: 0.9122807017543859
```

```
In [24]: plt.subplots(1, figsize=(7,5))
         plt.title('Decision Tree classifier')
         plt.xlabel("FPR")
         plt.ylabel("TPR")
         plt.yticks(tpr)
         plt.xticks(fpr)
         plt.plot(fpr,tpr)

Out[24]: [<matplotlib.lines.Line2D at 0x7f97f819ae80>]
```



TextEdit

Running the SVM algorithm.

In [25]:
```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
import sklearn.metrics as metrics

pipeline = Pipeline([('scaler',StandardScaler()),('classifier',SVC(kernel='rbf'))])
pipeline.fit(x_train,y_train)
pred_svm=pipeline.predict(x_test)
fpr,tpr,threshold=metrics.roc_curve(y_test,pred_svm)
roc_auc_rbf=metrics.auc(fpr,tpr)
```

In [26]:
```
roc_auc_rbf
```

Out[26]:
```
0.9787234042553192
```

In [27]:
```
accuracy_score(y_test,pred_svm)
```

Out[27]:
```
0.9824561403508771
```

In [28]:
```
plt.title("SVM")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.yticks(tpr)
plt.xticks(fpr)
plt.plot(fpr,tpr)
```

Out[28]:
```
[<matplotlib.lines.Line2D at 0x7f97e810c2b0>]
```