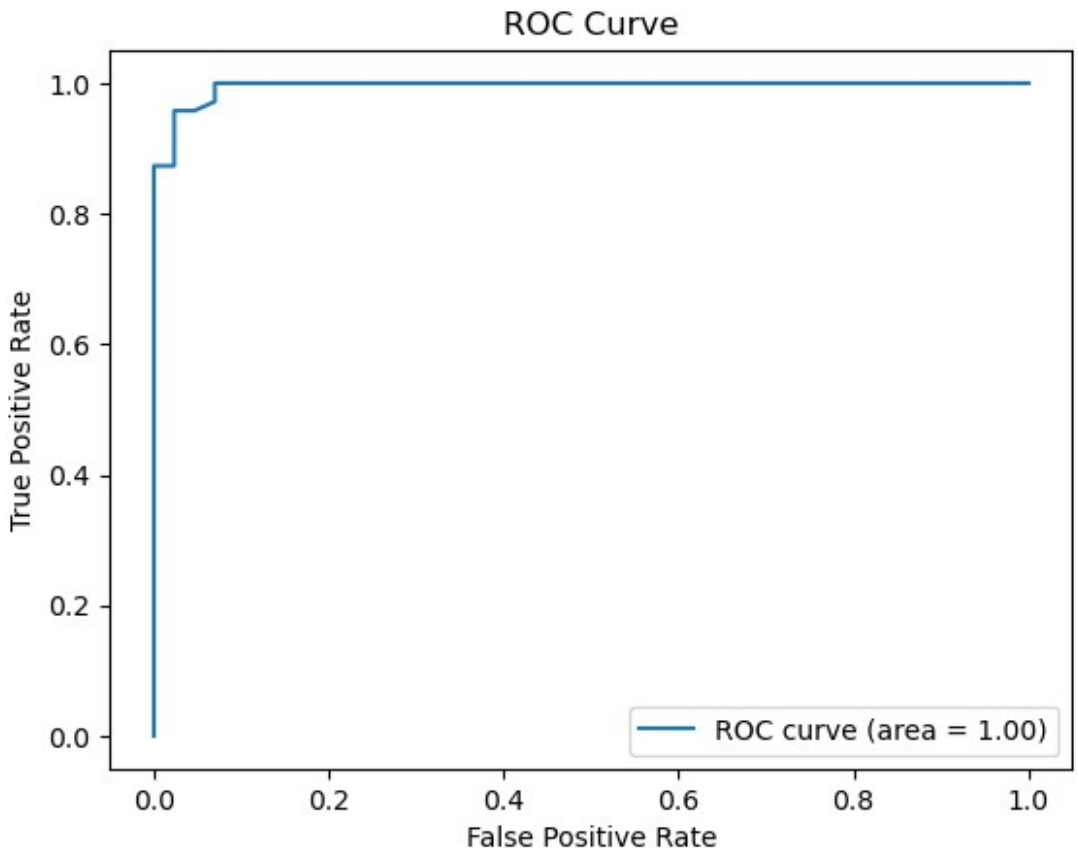
- Name: Deep Salunkhe
- Roll No.:21102A0014
- SEM-7 ML Lab4 Github Link

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer

# Load the dataset
data = load_breast_cancer()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score,
recall_score, f1_score, roc_curve, auc
import matplotlib.pyplot as plt

# Construct and train the classifier
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = clf.predict(X_test)

# Evaluate performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

# Print metrics
print(f"Accuracy: {accuracy}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1-Score: {f1}")

# Plot ROC Curve
y_prob = clf.predict_proba(X_test)[:, 1]
fpr, tpr, _ = roc_curve(y_test, y_prob)
roc_auc = auc(fpr, tpr)

plt.figure()
plt.plot(fpr, tpr, label=f'ROC curve (area = {roc_auc:.2f})')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
```

```
plt.legend(loc="lower right")
plt.show()
```

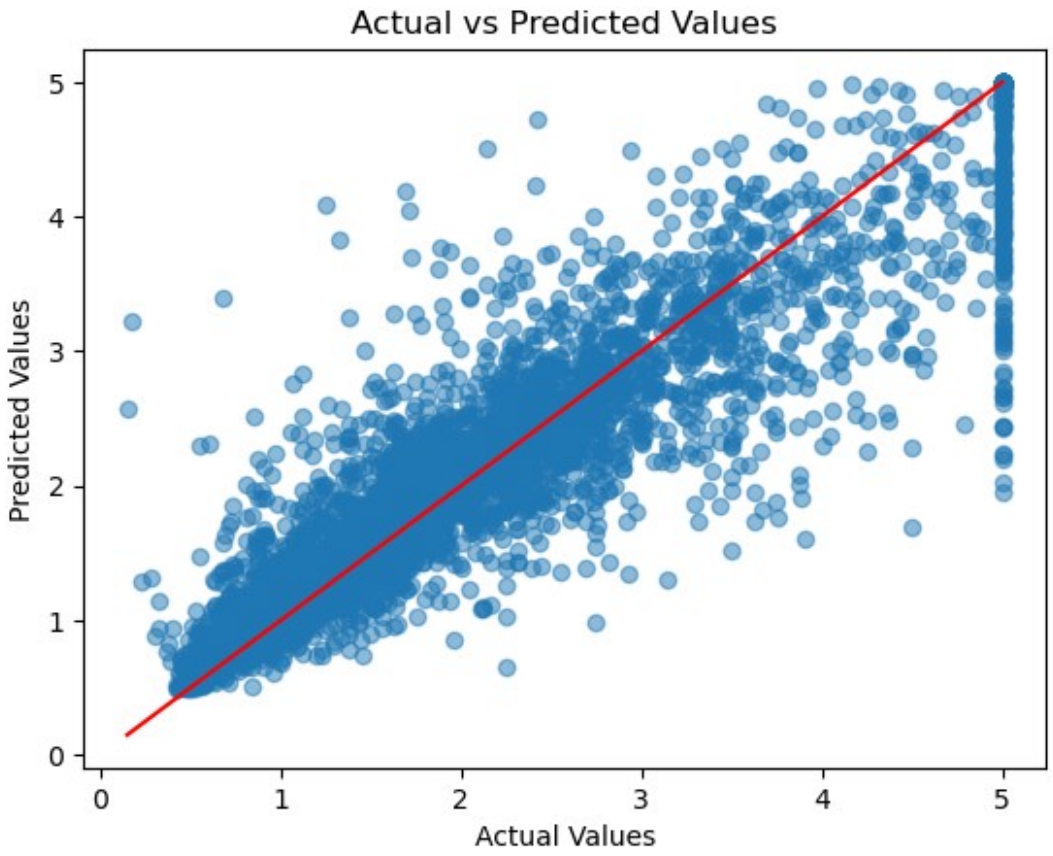
```
Accuracy: 0.9649122807017544
Precision: 0.958904109589041
Recall: 0.9859154929577465
F1-Score: 0.9722222222222222
```




```
from sklearn.datasets import fetch_california_housing

# Load the dataset
data = fetch_california_housing()
X = pd.DataFrame(data.data, columns=data.feature_names)
y = pd.Series(data.target)

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
```

```
# Construct and train the regressor
reg = RandomForestRegressor(random_state=42)
reg.fit(X_train, y_train)

# Predict the target variable on the testing set
y_pred = reg.predict(X_test)

# Evaluate performance
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print metrics
print(f"Mean Squared Error: {mse}")
print(f"R-Squared: {r2}")

# Plot actual vs predicted values
plt.figure()
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.title('Actual vs Predicted Values')
plt.show()
```

```
Mean Squared Error: 0.2553684927247781
R-Squared: 0.8051230593157366
```



1. Random Forest Classifier (Breast Cancer Wisconsin Dataset) Performance Metrics:

Accuracy: 0.9649 Precision: 0.9589 Recall: 0.9859 F1-Score: 0.9722 AUC: (Not directly provided, but implied from ROC curve analysis) Interpretation:

High Accuracy: The model correctly classifies around 96.5% of the instances, which indicates strong overall performance. Precision: About 95.9% of the positive predictions (malignant) made by the model are correct. This is crucial in a medical context where false positives can lead to unnecessary anxiety or treatment. Recall: The model identifies approximately 98.6% of actual malignant cases, which is essential for minimizing false negatives (i.e., missing a positive case). F1-Score: The balanced F1-score of 0.9722 reflects that the model performs well in both precision and recall, making it reliable in detecting malignancies.

1. Random Forest Regressor (California Housing Dataset) Performance Metrics:

Mean Squared Error (MSE): 0.2554 R-Squared (R^2): 0.8051 Interpretation:

MSE: The MSE of 0.2554 suggests that, on average, the squared difference between the actual and predicted house prices is relatively low, indicating that the model predicts house prices with reasonable accuracy. R-Squared: An R^2 value of 0.8051 indicates that approximately 80.5% of the variance in the house prices is explained by the model. This is a good result, showing that the model effectively captures the relationships in the data. Comparison and Insights Model Performance:

Classification Model: The Random Forest classifier exhibits strong performance with high accuracy, precision, recall, and F1-score. The model is particularly reliable in a critical application like medical diagnosis. Regression Model: The Random Forest regressor also performs well, with a good R^2 value and low MSE, making it suitable for predicting continuous outcomes like housing prices. Strengths:

Both models benefit from the inherent advantages of Random Forest, such as robustness to overfitting, ability to handle large datasets, and good performance even with imbalanced data. The classification model's high recall is particularly valuable in ensuring that few malignant cases are missed. Weaknesses:

While both models perform well, the regressor's R^2 value of 0.8051, though good, indicates that there's still room for improvement in capturing the variability in house prices. The classifier's precision, while high, leaves a slight room for improvement in reducing false positives. Recommendations:

For the Classification Model: Fine-tuning the model parameters (e.g., increasing the number of trees, adjusting the maximum depth) could further improve precision without sacrificing recall. For the Regression Model: Experimenting with additional features or tuning hyperparameters might reduce the MSE and improve the R^2 value. Conclusion Both models demonstrate strong performance in their respective tasks, with the classifier being particularly well-suited for high-stakes scenarios like medical diagnostics, while the regressor provides reliable predictions for housing prices. Fine-tuning and further experimentation could enhance the models' performance even further.