Artificial Intelligence Project Report

Course Name: CSC4101 - Artificial Intelligence
Course Instructor: Sheikh Usama Khalid

Course Name: CSCL4101 - Lab Artificial Intelligence
Course Instructor: Wali Muhammad Khubaib

Group Member Reg No:

- Muhammad Taha Jamal-2112244
- Mazahir Abbas-2112244

Table of Content

Introduction

This report focuses on the analysis and prediction of breast cancer using a dataset containing various features indicative of benign and malignant tumors. Accurate diagnosis is crucial for appropriate treatment planning and patient care, and predictive models can help in early detection and prognosis of the disease. We utilized several machine learning algorithms to classify tumor samples as benign or malignant.

Dataset Overview

The dataset consists of 30 features representing characteristics of cell nuclei present in digitized images of breast mass. The target variable is a binary indicator (diagnosis), where 1 denotes malignant and 0 indicates benign. Here is a brief overview of some key features:

- 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)

Machine Learning Algorithms

K-Nearest Neighbors (KNN)

- **Overview:** Classifies samples based on the majority class among its k-nearest neighbors.
- **Advantages:** Simple and effective for small datasets.
- **Disadvantages:** Sensitive to the scale of data and irrelevant features.

Decision Tree

- **Overview:** Uses a tree-like model of decisions and their possible consequences.
- **Advantages:** Easy to interpret and understand.
- **Disadvantages:** Can create over-complex trees that do not generalize well.

Random Forest

- **Overview:** An ensemble of Decision Trees, typically trained via the bagging method.
- **Advantages:** Reduction in overfitting and random forest is more accurate than decision trees in most cases.

- **Disadvantages:** Slow real-time prediction, complex, and difficult to implement.

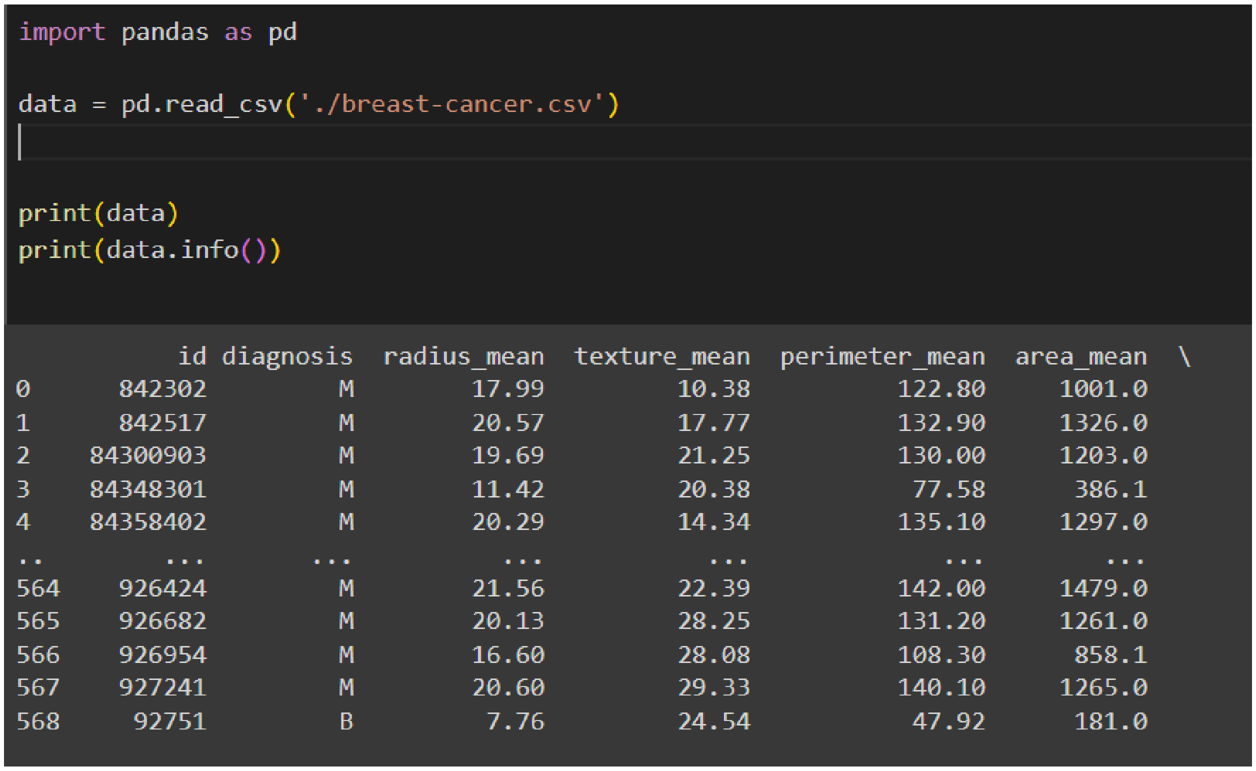
Cascading Classifier

- **Overview:** Combines the predictions of a KNN model with a Random Forest model for enhanced prediction accuracy.
- **Advantages:** Improves prediction robustness.
- **Disadvantages:** Increases computational cost.

Code Implementation and Output

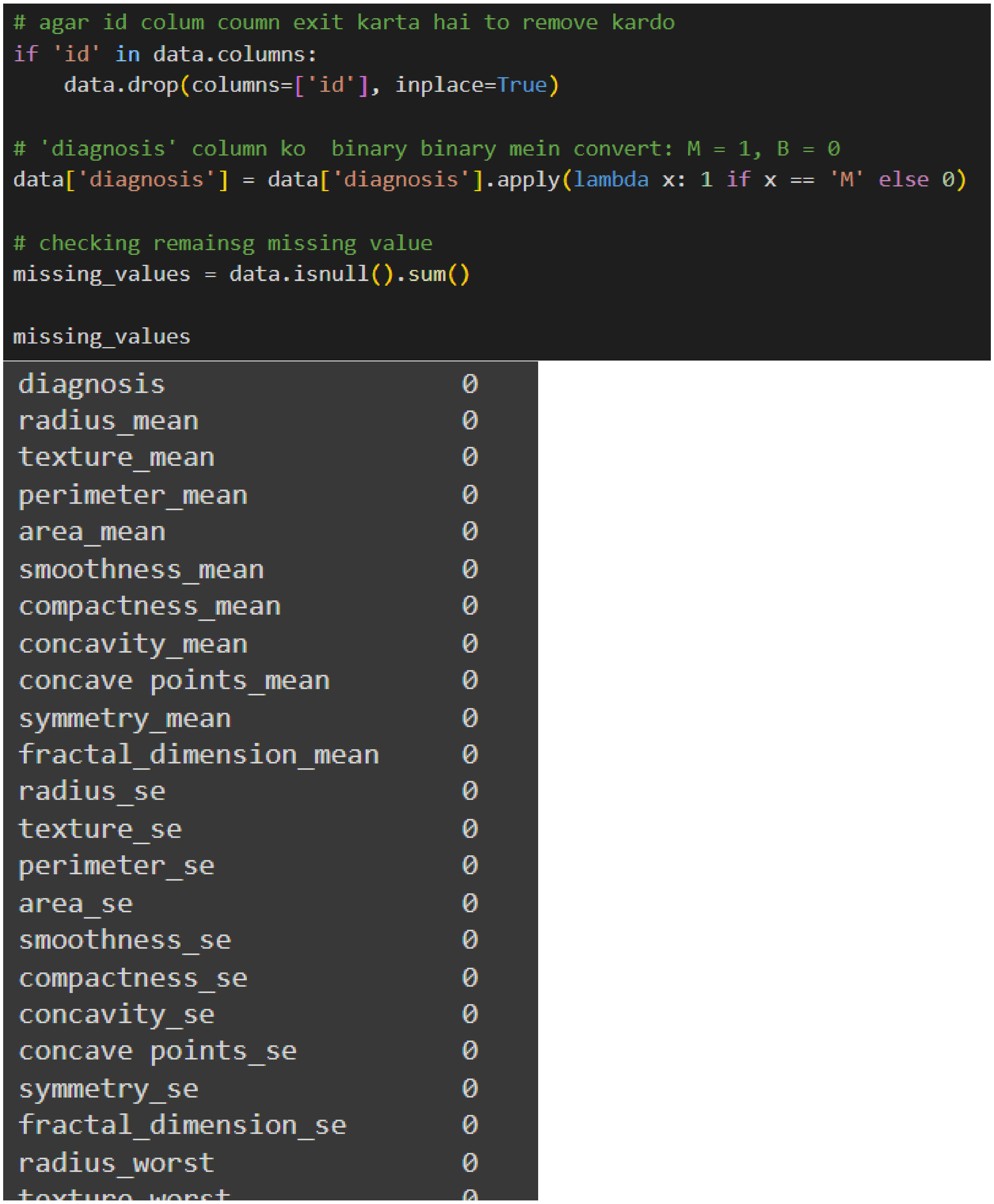
- **K-Nearest Neighbors (KNN):** Achieved an accuracy of 94.74%.
- **Decision Tree:** Achieved an accuracy of 93.86%.
- **Random Forest:** Achieved an accuracy of 96.49%.
- **Cascading Classifier:** Achieved an accuracy of 95.61%.

Code Screen Shots:



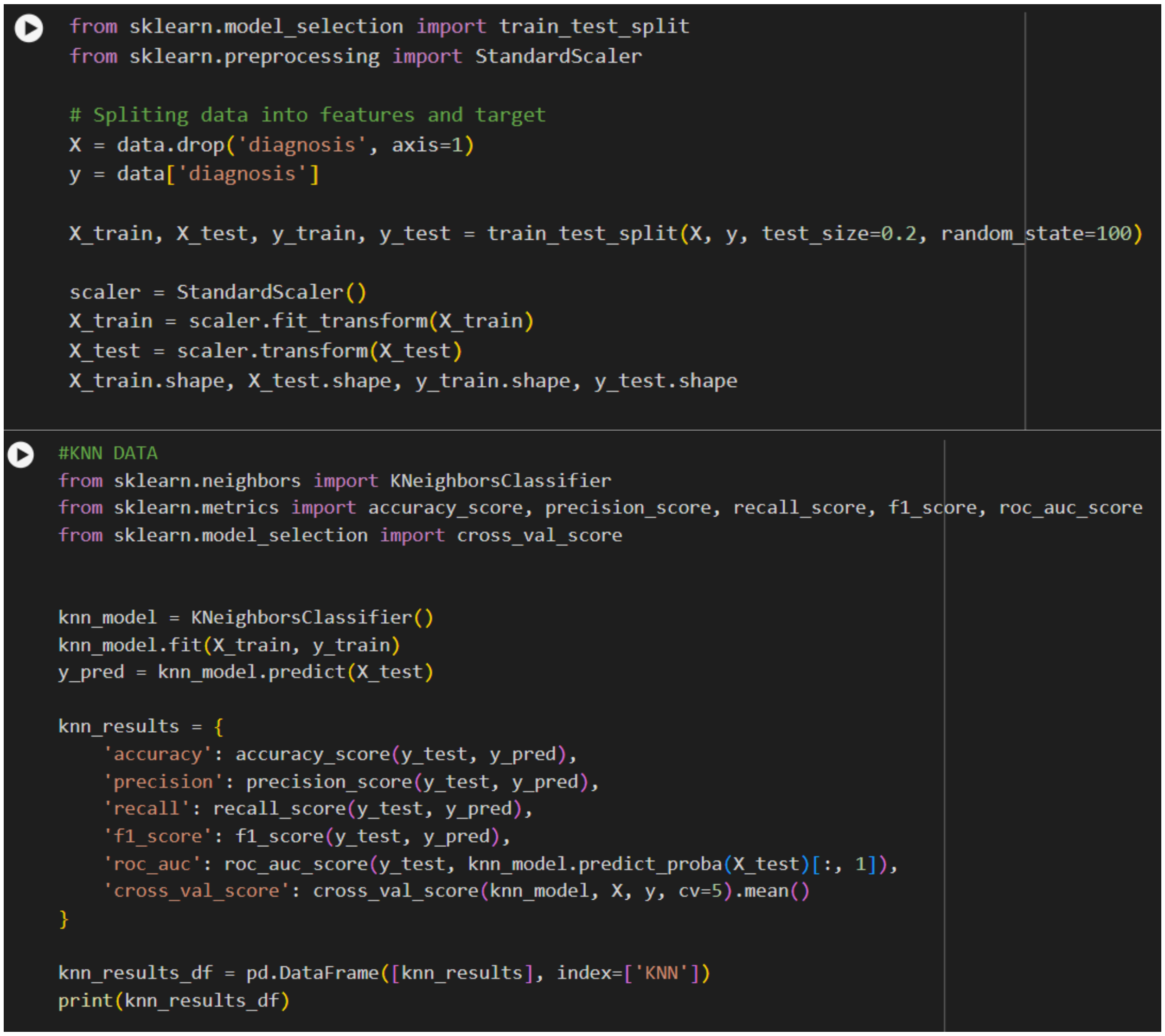
```python
import pandas as pd

data = pd.read_csv('./breast-cancer.csv')


print(data)
print(data.info())
```

```
           id diagnosis  radius_mean  texture_mean  perimeter_mean  area_mean  \
0      842302         M        17.99         10.38          122.80     1001.0
1      842517         M        20.57         17.77          132.90     1326.0
2    84300903         M        19.69         21.25          130.00     1203.0
3    84348301         M        11.42         20.38           77.58      386.1
4    84358402         M        20.29         14.34          135.10     1297.0
..        ...       ...          ...           ...             ...        ...
564    926424         M        21.56         22.39          142.00     1479.0
565    926682         M        20.13         28.25          131.20     1261.0
566    926954         M        16.60         28.08          108.30      858.1
567    927241         M        20.60         29.33          140.10     1265.0
568     92751         B         7.76         24.54           47.92      181.0
```



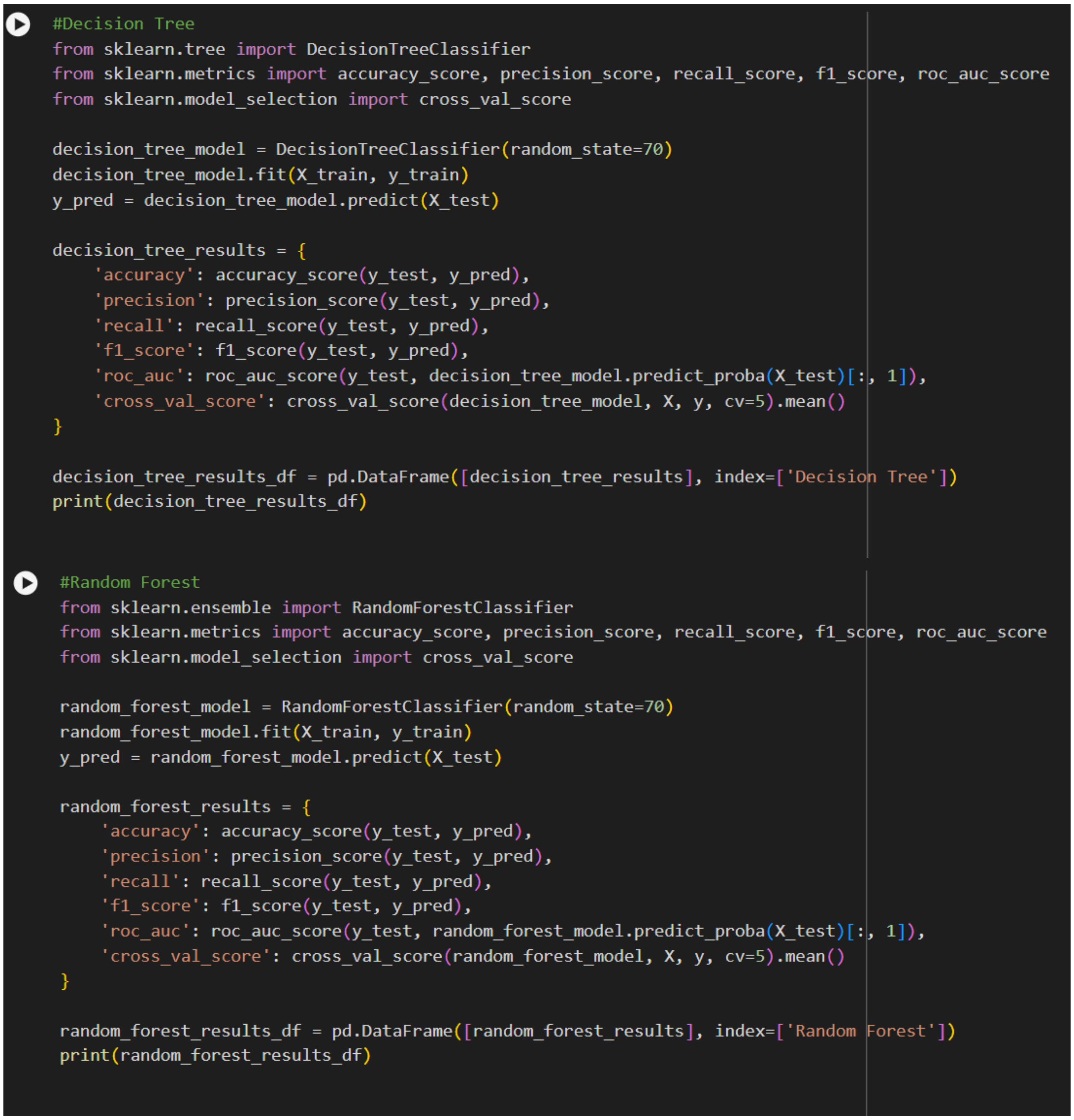
```python
# agar id colum coumn exit karta hai to remove kardo
if 'id' in data.columns:
    data.drop(columns=['id'], inplace=True)

# 'diagnosis' column ko  binary binary mein convert: M = 1, B = 0
data['diagnosis'] = data['diagnosis'].apply(lambda x: 1 if x == 'M' else 0)

# checking remainsg missing value
missing_values = data.isnull().sum()

missing_values
```

```
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
```



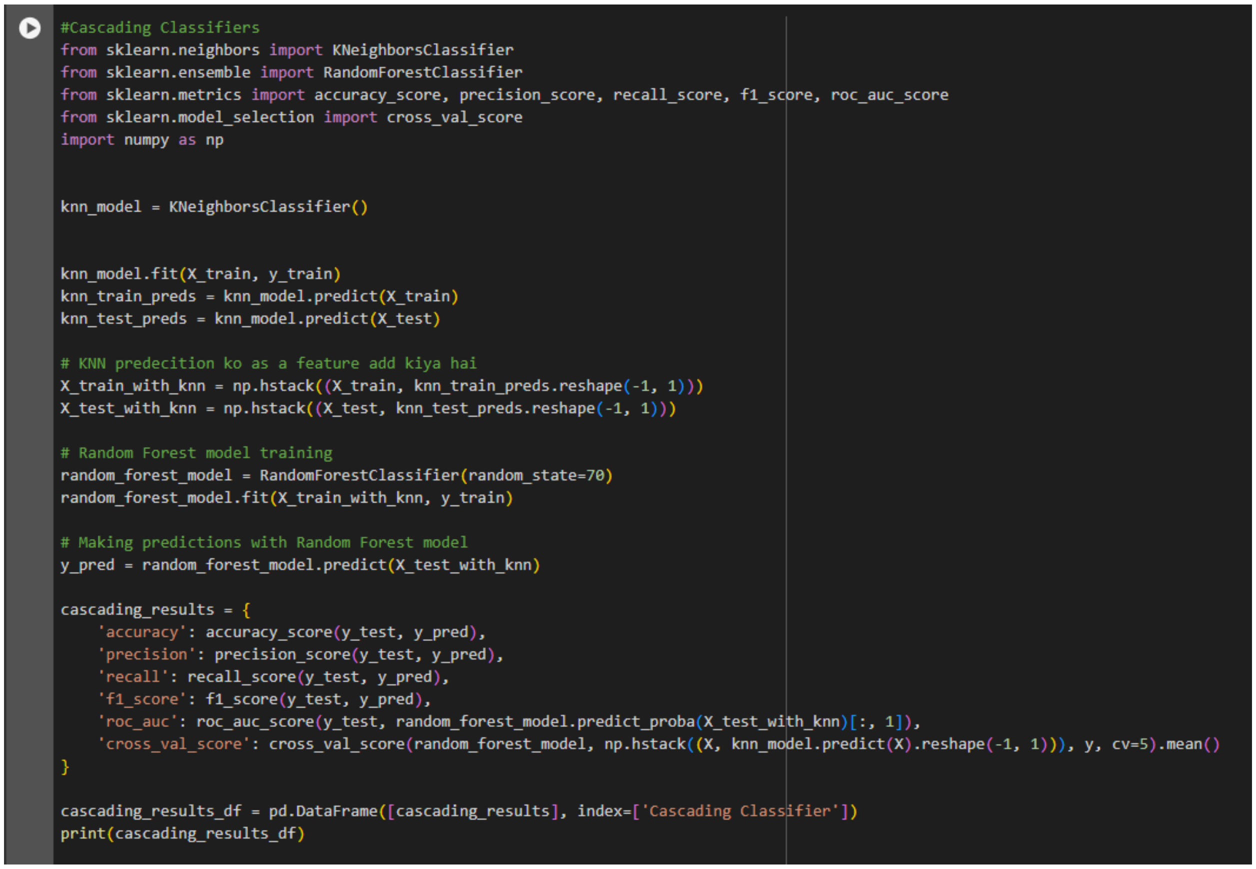
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Spliting data into features and target
X = data.drop('diagnosis', axis=1)
y = data['diagnosis']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=100)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

```
#KNN DATA
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import cross_val_score


knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)
y_pred = knn_model.predict(X_test)

knn_results = {
    'accuracy': accuracy_score(y_test, y_pred),
    'precision': precision_score(y_test, y_pred),
    'recall': recall_score(y_test, y_pred),
    'f1_score': f1_score(y_test, y_pred),
    'roc_auc': roc_auc_score(y_test, knn_model.predict_proba(X_test)[:, 1]),
    'cross_val_score': cross_val_score(knn_model, X, y, cv=5).mean()
}

knn_results_df = pd.DataFrame([knn_results], index=['KNN'])
print(knn_results_df)
```



```
#Decision Tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import cross_val_score

decision_tree_model = DecisionTreeClassifier(random_state=70)
decision_tree_model.fit(X_train, y_train)
y_pred = decision_tree_model.predict(X_test)

decision_tree_results = {
    'accuracy': accuracy_score(y_test, y_pred),
    'precision': precision_score(y_test, y_pred),
    'recall': recall_score(y_test, y_pred),
    'f1_score': f1_score(y_test, y_pred),
    'roc_auc': roc_auc_score(y_test, decision_tree_model.predict_proba(X_test)[:, 1]),
    'cross_val_score': cross_val_score(decision_tree_model, X, y, cv=5).mean()
}

decision_tree_results_df = pd.DataFrame([decision_tree_results], index=['Decision Tree'])
print(decision_tree_results_df)
```

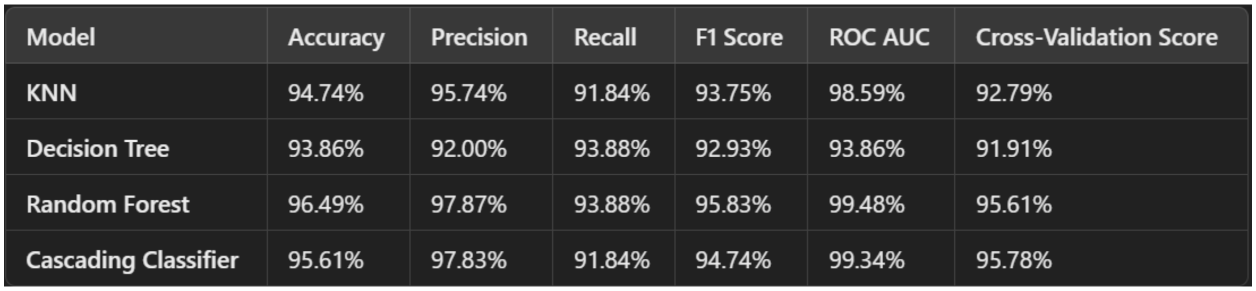
```
#Random Forest
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import cross_val_score

random_forest_model = RandomForestClassifier(random_state=70)
random_forest_model.fit(X_train, y_train)
y_pred = random_forest_model.predict(X_test)

random_forest_results = {
    'accuracy': accuracy_score(y_test, y_pred),
    'precision': precision_score(y_test, y_pred),
    'recall': recall_score(y_test, y_pred),
    'f1_score': f1_score(y_test, y_pred),
    'roc_auc': roc_auc_score(y_test, random_forest_model.predict_proba(X_test)[:, 1]),
    'cross_val_score': cross_val_score(random_forest_model, X, y, cv=5).mean()
}

random_forest_results_df = pd.DataFrame([random_forest_results], index=['Random Forest'])
print(random_forest_results_df)
```



```python
#Cascading Classifiers
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
from sklearn.model_selection import cross_val_score
import numpy as np


knn_model = KNeighborsClassifier()


knn_model.fit(X_train, y_train)
knn_train_preds = knn_model.predict(X_train)
knn_test_preds = knn_model.predict(X_test)

# KNN predecition ko as a feature add kiya hai
X_train_with_knn = np.hstack((X_train, knn_train_preds.reshape(-1, 1)))
X_test_with_knn = np.hstack((X_test, knn_test_preds.reshape(-1, 1)))

# Random Forest model training
random_forest_model = RandomForestClassifier(random_state=70)
random_forest_model.fit(X_train_with_knn, y_train)

# Making predictions with Random Forest model
y_pred = random_forest_model.predict(X_test_with_knn)

cascading_results = {
    'accuracy': accuracy_score(y_test, y_pred),
    'precision': precision_score(y_test, y_pred),
    'recall': recall_score(y_test, y_pred),
    'f1_score': f1_score(y_test, y_pred),
    'roc_auc': roc_auc_score(y_test, random_forest_model.predict_proba(X_test_with_knn)[:, 1]),
    'cross_val_score': cross_val_score(random_forest_model, np.hstack((X, knn_model.predict(X).reshape(-1, 1))), y, cv=5).mean()
}

cascading_results_df = pd.DataFrame([cascading_results], index=['Cascading Classifier'])
print(cascading_results_df)
```

Conclusion

Model Performance Metrics



Model	Accuracy	Precision	Recall	F1 Score	ROC AUC	Cross-Validation Score
KNN	94.74%	95.74%	91.84%	93.75%	98.59%	92.79%
Decision Tree	93.86%	92.00%	93.88%	92.93%	93.86%	91.91%
Random Forest	96.49%	97.87%	93.88%	95.83%	99.48%	95.61%
Cascading Classifier	95.61%	97.83%	91.84%	94.74%	99.34%	95.78%

Analysis

- Random Forest performs exceptionally well across all metrics, particularly in ROC AUC and accuracy, making it a strong candidate for this type of classification problem.
- KNN also shows strong results, especially in terms of ROC AUC and precision.
- Decision Tree performs slightly lower than the others but still delivers robust results, especially in recall.
- The Cascading Classifier improves slightly on the cross-validation score compared to a standalone Random Forest, suggesting it might generalize better despite a marginal decrease in recall.

Based on these results, Random Forest and the Cascading Classifier are the most effective models for this dataset, with high scores in both accuracy and robustness as indicated by cross-validation and ROC AUC scores.