untitled4

June 28, 2024

[6]:

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, mean_squared_error
from google.colab import drive

# Mount Google Drive
drive.mount('/content/drive')

# Load the cancer dataset from Google Drive
data_cancer = pd.read_csv("/content/drive/MyDrive/cancer_dataset/Cancer_Data.
 ↪csv")
print(data_cancer.head())

# Hypothetical continuous target variable 'Tumor_Size'
# Assuming 'Tumor_Size' is in the dataset
# Here, I will create a synthetic 'Tumor_Size' column for demonstration purposes
np.random.seed(42)
data_cancer['Tumor_Size'] = np.random.uniform(1.0, 5.0, size=len(data_cancer))

# Select features and target for regression
X = data_cancer[['radius_mean', 'area_mean']]
y_classification = data_cancer['diagnosis']
y_regression = data_cancer['Tumor_Size']

# Split data into training and testing sets
X_train, X_test, y_train_classification, y_test_classification =␣
 ↪train_test_split(X, y_classification, test_size=0.2, random_state=42)
_, _, y_train_regression, y_test_regression = train_test_split(X, y_regression,␣
 ↪test_size=0.2, random_state=42)

# K-NN Classifier
k = 3
```

```python
knn = KNeighborsClassifier(n_neighbors=k)
knn.fit(X_train, y_train_classification)

# Logistic Regression Classifier
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train_classification)

# Decision Tree Classifier
decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(X_train, y_train_classification)

# Random Forest Classifier
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(X_train, y_train_classification)

# Linear Regression Model
linear_model = LinearRegression()
linear_model.fit(X_train, y_train_regression)

# Get user input
radius_mean = float(input("Enter radius mean: "))
area_mean = float(input("Enter area mean: "))

# Prepare new data for prediction
new_data = np.array([[radius_mean, area_mean]])

# K-NN Prediction
knn_prediction = knn.predict(new_data)

# Logistic Regression Prediction
logistic_prediction = logistic_model.predict(new_data)

# Decision Tree Prediction
tree_prediction = decision_tree.predict(new_data)

# Random Forest Prediction
forest_prediction = random_forest.predict(new_data)

# Linear Regression Prediction
linear_prediction = linear_model.predict(new_data)

# Output predictions
if knn_prediction[0] == 'M':
    print("K-NN prediction: cancer is present")
else:
    print("K-NN prediction: there is no cancer")
```

```
if logistic_prediction[0] == 'M':
    print("Logistic Regression prediction: cancer is present")
else:
    print("Logistic Regression prediction: there is no cancer")

if tree_prediction[0] == 'M':
    print("Decision Tree prediction: cancer is present")
else:
    print("Decision Tree prediction: there is no cancer")

if forest_prediction[0] == 'M':
    print("Random Forest prediction: cancer is present")
else:
    print("Random Forest prediction: there is no cancer")

print(f"Linear Regression prediction for Tumor_Size: {linear_prediction[0]:.
 ↪2f}")

# Evaluate models
y_pred_knn = knn.predict(X_test)
knn_accuracy = accuracy_score(y_test_classification, y_pred_knn)
print(f"K-NN Model accuracy: {knn_accuracy:.2f}")

y_pred_logistic = logistic_model.predict(X_test)
logistic_accuracy = accuracy_score(y_test_classification, y_pred_logistic)
print(f"Logistic Regression Model accuracy: {logistic_accuracy:.2f}")

y_pred_tree = decision_tree.predict(X_test)
tree_accuracy = accuracy_score(y_test_classification, y_pred_tree)
print(f"Decision Tree Model accuracy: {tree_accuracy:.2f}")

y_pred_forest = random_forest.predict(X_test)
forest_accuracy = accuracy_score(y_test_classification, y_pred_forest)
print(f"Random Forest Model accuracy: {forest_accuracy:.2f}")

y_pred_linear = linear_model.predict(X_test)
linear_mse = mean_squared_error(y_test_regression, y_pred_linear)
print(f"Linear Regression Model MSE: {linear_mse:.2f}")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
         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
```

```
   smoothness_mean  compactness_mean  concavity_mean  concave points_mean  \
0          0.11840           0.27760          0.3001              0.14710
1          0.08474           0.07864          0.0869              0.07017
2          0.10960           0.15990          0.1974              0.12790
3          0.14250           0.28390          0.2414              0.10520
4          0.10030           0.13280          0.1980              0.10430

   …  texture_worst  perimeter_worst  area_worst  smoothness_worst  \
0  …          17.33           184.60      2019.0            0.1622
1  …          23.41           158.80      1956.0            0.1238
2  …          25.53           152.50      1709.0            0.1444
3  …          26.50            98.87       567.7            0.2098
4  …          16.67           152.20      1575.0            0.1374

   compactness_worst  concavity_worst  concave points_worst  symmetry_worst  \
0             0.6656           0.7119                0.2654          0.4601
1             0.1866           0.2416                0.1860          0.2750
2             0.4245           0.4504                0.2430          0.3613
3             0.8663           0.6869                0.2575          0.6638
4             0.2050           0.4000                0.1625          0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
Enter radius mean: 17.99
Enter area mean: 1001.0
K-NN prediction: cancer is present
Logistic Regression prediction: cancer is present
Decision Tree prediction: cancer is present
Random Forest prediction: cancer is present
Linear Regression prediction for Tumor_Size: 2.94
K-NN Model accuracy: 0.89
Logistic Regression Model accuracy: 0.92
Decision Tree Model accuracy: 0.83
Random Forest Model accuracy: 0.91
Linear Regression Model MSE: 1.57
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but KNeighborsClassifier was fitted with feature
names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
```

```
not have valid feature names, but LogisticRegression was fitted with feature
names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but DecisionTreeClassifier was fitted with feature
names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but RandomForestClassifier was fitted with feature
names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```