```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn.metrics import mean_squared_error, r2_score,
accuracy_score, confusion_matrix, classification_report
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor, plot_tree
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt

# Load the dataset
file_path = '/content/drive/MyDrive/student_admission_dataset.csv'  #
Update this path accordingly
df = pd.read_csv(file_path)

# Define independent variables (features) and dependent variable
(target)
X = df[['GPA', 'SAT_Score']]
y = df['Admission_Status']

# Encode the categorical target variable
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)

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

# Linear Regression
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)
mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)
print("Linear Regression - Mean Squared Error:", mse_linear)
print("Linear Regression - R-squared:", r2_linear)
print("Linear Regression - Coefficients:", linear_model.coef_)
print("Linear Regression - Intercept:", linear_model.intercept_)
```

```
Linear Regression - Mean Squared Error: 0.6292512036814438
Linear Regression - R-squared: 0.006863630553276767
Linear Regression - Coefficients: [ 0.12239976 -0.00045621]
Linear Regression - Intercept: 1.1901089445004878
```

```
# Logistic Regression
logistic_model = LogisticRegression()
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
```

```
conf_matrix_logistic = confusion_matrix(y_test, y_pred_logistic)
class_report_logistic = classification_report(y_test, y_pred_logistic,
target_names=label_encoder.classes_)
print("Logistic Regression - Accuracy:", accuracy_logistic)
print("Logistic Regression - Confusion Matrix:\n",
conf_matrix_logistic)
print("Logistic Regression - Classification Report:\n",
class_report_logistic)

# Predicting the admission status of a new student with specific
features
new_student = pd.DataFrame({'GPA': [3.5], 'SAT_Score': [1200]})
predicted_admission_linear = linear_model.predict(new_student)
predicted_admission_logistic = logistic_model.predict(new_student)
print("Predicted Admission for new student (Linear Regression -
encoded):", predicted_admission_linear[0])
print("Predicted Admission for new student (Linear Regression -
decoded):",
label_encoder.inverse_transform([int(predicted_admission_linear[0])]))
print("Predicted Admission for new student (Logistic Regression -
encoded):", predicted_admission_logistic[0])
print("Predicted Admission for new student (Logistic Regression -
decoded):",
label_encoder.inverse_transform([predicted_admission_logistic[0]]))
```

```
Logistic Regression - Accuracy: 0.3
Logistic Regression - Confusion Matrix:
 [[6 2 6]
 [7 6 5]
 [7 8 3]]
Logistic Regression - Classification Report:
              precision    recall  f1-score   support

    Accepted       0.30      0.43      0.35        14
    Rejected       0.38      0.33      0.35        18
  Waitlisted       0.21      0.17      0.19        18

    accuracy                           0.30        50
   macro avg       0.30      0.31      0.30        50
weighted avg       0.30      0.30      0.29        50

Predicted Admission for new student (Linear Regression - encoded):
1.0710542915521117
Predicted Admission for new student (Linear Regression - decoded):
['Rejected']
Predicted Admission for new student (Logistic Regression - encoded): 1
Predicted Admission for new student (Logistic Regression - decoded):
['Rejected']
```

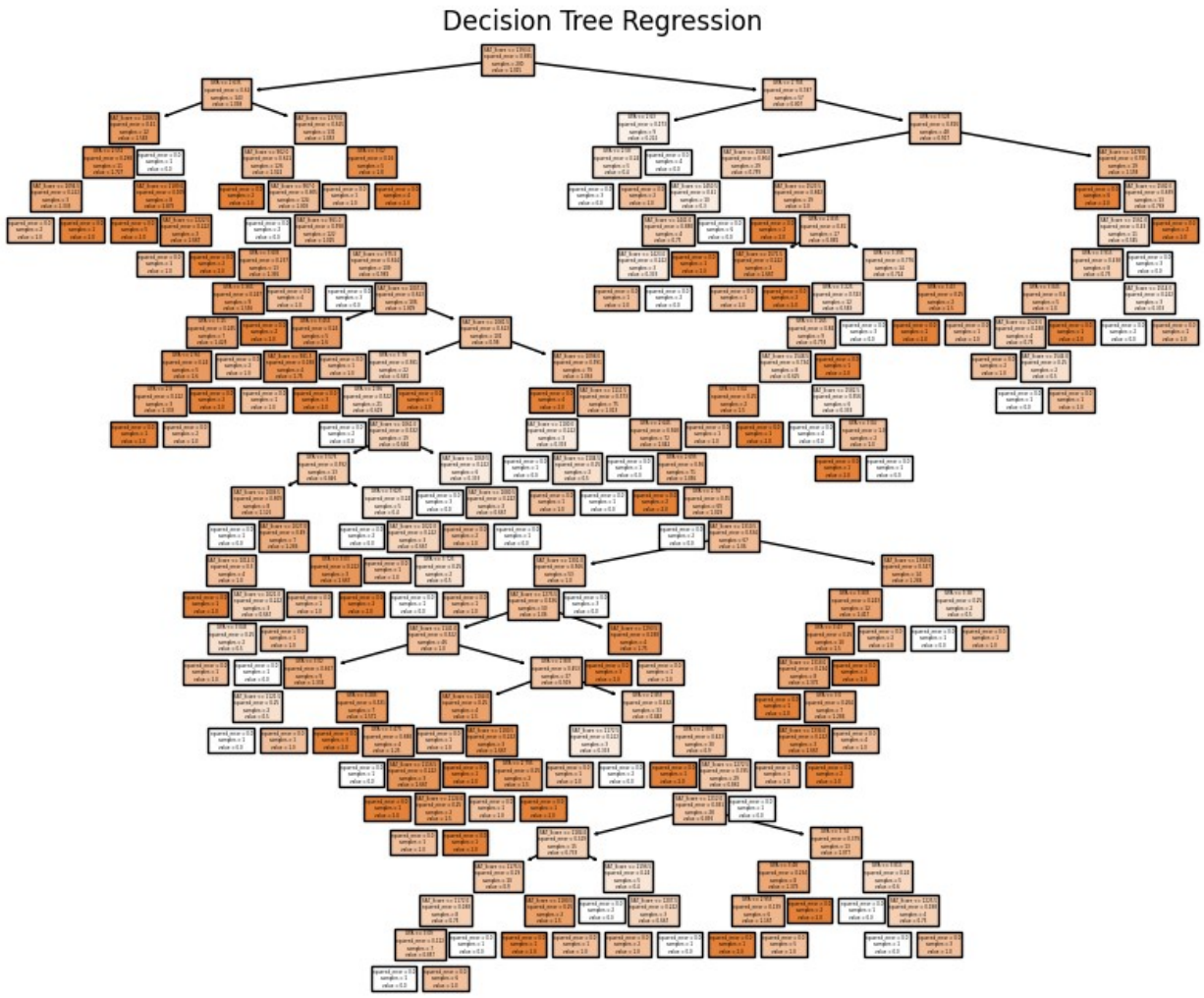
```
# K-Nearest Neighbors (K-NN) Regressor
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
k = 3
knn_regressor = KNeighborsRegressor(n_neighbors=k)
knn_regressor.fit(X_train_scaled, y_train)
y_pred_knn = knn_regressor.predict(X_test_scaled)
rmse_knn = mean_squared_error(y_test, y_pred_knn, squared=False)
r2_knn = r2_score(y_test, y_pred_knn)
print(f'K-NN Regressor - Root Mean Squared Error (RMSE): {rmse_knn}')
print(f'K-NN Regressor - R^2 Score: {r2_knn}')
new_student_scaled = scaler.transform([[3.5, 1200]])
predicted_admission_knn = knn_regressor.predict(new_student_scaled)
print(f'Predicted Admission (K-NN Regressor - encoded):
{predicted_admission_knn[0]}')
print("Predicted Admission for new student (K-NN Regressor -
decoded):",
label_encoder.inverse_transform([int(predicted_admission_knn[0])]))
```

```
K-NN Regressor - Root Mean Squared Error (RMSE): 0.8432740427115678
K-NN Regressor - R^2 Score: -0.122334455667789
Predicted Admission (K-NN Regressor - encoded): 1.0
Predicted Admission for new student (K-NN Regressor - decoded):
['Rejected']
```

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

```
# Decision Tree Regressor
dt_regressor = DecisionTreeRegressor(random_state=42)
dt_regressor.fit(X_train, y_train)
y_pred_dt = dt_regressor.predict(X_test)
mse_dt = mean_squared_error(y_test, y_pred_dt)
r2_dt = r2_score(y_test, y_pred_dt)
print(f'Decision Tree Regressor - Mean Squared Error (MSE): {mse_dt}')
print(f'Decision Tree Regressor - R-squared (R2): {r2_dt}')
plt.figure(figsize=(10, 8))
plot_tree(dt_regressor, feature_names=X.columns, filled=True)
plt.title("Decision Tree Regression")
plt.show()
```

```
Decision Tree Regressor - Mean Squared Error (MSE): 0.94
Decision Tree Regressor - R-squared (R2): -0.48358585858585856
```




```
# Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100,
random_state=42)
rf_regressor.fit(X_train, y_train)
y_pred_rf = rf_regressor.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)
print(f'Random Forest Regressor - Mean Squared Error (MSE): {mse_rf}')
print(f'Random Forest Regressor - R-squared (R2): {r2_rf}')
feature_importances = rf_regressor.feature_importances_
features = X.columns
plt.figure(figsize=(8, 6))
plt.bar(features, feature_importances, color='skyblue')
plt.xlabel('Features')
plt.ylabel('Importance')
plt.title('Feature Importance in Random Forest Model')
plt.show()
```

```
Random Forest Regressor - Mean Squared Error (MSE): 0.6992299999999999
Random Forest Regressor - R-squared (R2): -0.10358270202020181
```