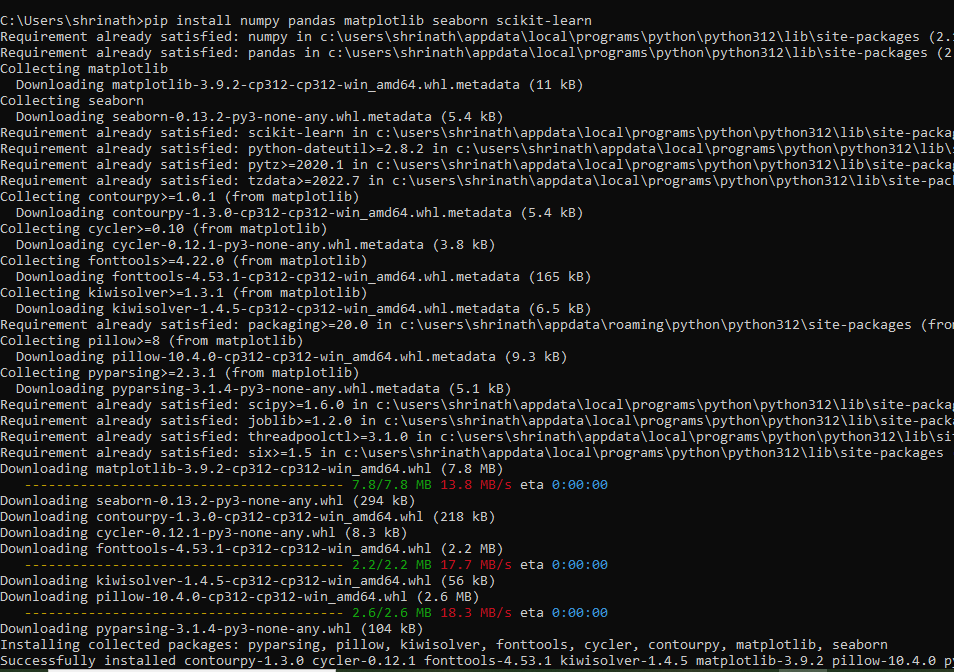
# ASSIGNMENT-1 MACHINE LEARNING

**IMPLEMENTATION USING MACHINE LEARNING ALGORTHIMS IN PYTHON**

1. **CLASSFICATION ALGORTHIM -2**
2. **FUTURE ALGORTHIM -2**
3. **COMPARE THIS TWO ALGORTHIM WITH GRAPH WHICH IS BEST.**

**INSTALL PYTHON PACKAGES AND LIBARIES:**



**DATASET AS WE TAKEN AS BIKES**

**RANDOM VALUES FOR POWER,TORQUE,MILLEGE.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Bajaj Avenger Cruise 220 2017 | 2017 | 17,000 Km | First owner | Hyderabad | 35 kmpl | 19 bhp | 63,500 |
| Royal Enfield Classic 350cc 2016 | 2016 | 50,000 Km | First owner | Hyderabad | 35 kmpl | 19.80 bhp | 115,000 |
| Hyosung GT250R 2012 | 2012 | 14,795 Km | First owner | Hyderabad | 30 kmpl | 28 bhp | 300,000 |
| Bajaj Dominar 400 ABS 2017 | 2017 | 28 Km | First owner | Pondicherry | 28 kmpl | 34.50 bhp | 100,000 |
| Jawa Perak 330cc 2020 | 2020 | 2,000 Km | First owner | Bangalore |  | 30 bhp | 197,500 |
| KTM Duke 200cc 2012 | 2012 | 24,561 Km | Third owner | Bangalore | 35 kmpl | 25 bhp | 63,400 |
| Bajaj Pulsar 180cc 2016 | 2016 | 19,718 Km | First owner | Bangalore | 65 kmpl | 17 bhp | 55,000 |
| TVS Apache RTR 200 4V Dual Channel ABS BS6 2020 | 2020 | 40 kmpl | First owner | Hyderabad | 40 kmpl | 20.21 bhp | 120,000 |
| KTM Duke 390cc 2018 | 2018 | 1,350 Km | First owner | Jaipur | 25 kmpl | 42.90 bhp | 198,000 |

1. **CLASSFICATION ALGORTHIM:**
2. **K-NN**
3. **DESCISION TREE**

**CODE:**

**import pandas as pd**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**data = {**

**'Model\_Name': ['Bajaj Avenger Cruise 220', 'Royal Enfield Classic 350cc', 'Hyosung GT250R',**

**'Bajaj Dominar 400 ABS', 'Jawa Perak 330cc', 'KTM Duke 200cc', 'Bajaj Pulsar 180cc',**

**'TVS Apache RTR 200 4V', 'KTM Duke 390cc'],**

**'Model\_Year': [2017, 2016, 2012, 2017, 2020, 2012, 2016, 2020, 2018],**

**'Kms\_Driven': [17000, 50000, 14795, 28, 2000, 24561, 19718, 40, 1350],**

**'Owner': ['first owner', 'first owner', 'first owner', 'first owner', 'first owner', 'third owner',**

**'first owner', 'first owner', 'first owner'],**

**'Location': ['hyderabad', 'hyderabad', 'hyderabad', 'pondicherry', 'bangalore', 'bangalore',**

**'bangalore', 'hyderabad', 'jaipur'],**

**'Mileage': [35, 35, 30, 28, None, 35, 65, 40, 25],**

**'Power': [19, 19.8, 28, 34.5, 30, 25, 17, 20.21, 42.9],**

**'Price': [63500, 115000, 300000, 100000, 197500, 63400, 55000, 120000, 198000]**

**}**

**df = pd.DataFrame(data)**

**df['Owner'] = df['Owner'].map({'first owner': 1, 'second owner': 2, 'third owner': 3})**

**df['Location'] = df['Location'].map({'hyderabad': 1, 'pondicherry': 2, 'bangalore': 3, 'jaipur': 4})**

**df['Mileage'].fillna(df['Mileage'].mean(), inplace=True)**

**df['Category'] = df['Price'].apply(lambda x: 'Expensive' if x > 100000 else 'Affordable')**

**X = df[['Model\_Year', 'Kms\_Driven', 'Owner', 'Location', 'Mileage', 'Power']]**

**y = df['Category']**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)**

**dt\_classifier = DecisionTreeClassifier(random\_state=42)**

**dt\_classifier.fit(X\_train, y\_train)**

**dt\_predictions = dt\_classifier.predict(X\_test)**

**rf\_classifier = RandomForestClassifier(random\_state=42)**

**rf\_classifier.fit(X\_train, y\_train)**

**rf\_predictions = rf\_classifier.predict(X\_test)**

**plt.figure(figsize=(10, 5))**

**plt.subplot(1, 2, 1)**

**sns.heatmap(confusion\_matrix(y\_test, dt\_predictions), annot=True, fmt='d', cmap='Blues')**

**plt.title('Decision Tree Confusion Matrix')**

**plt.xlabel('Predicted')**

**plt.ylabel('Actual')**

**plt.subplot(1, 2, 2)**

**sns.heatmap(confusion\_matrix(y\_test, rf\_predictions), annot=True, fmt='d', cmap='Greens')**

**plt.title('Random Forest Confusion Matrix')**

**plt.xlabel('Predicted')**

**plt.ylabel('Actual')**

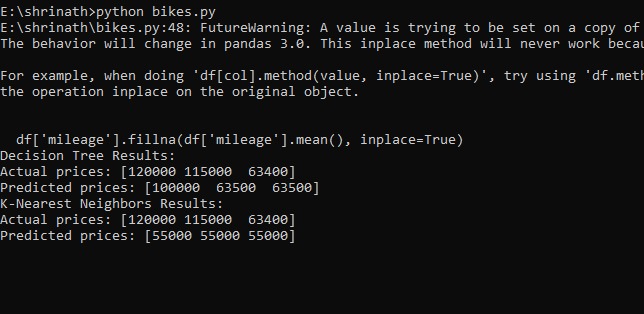
**plt.tight\_layout()**

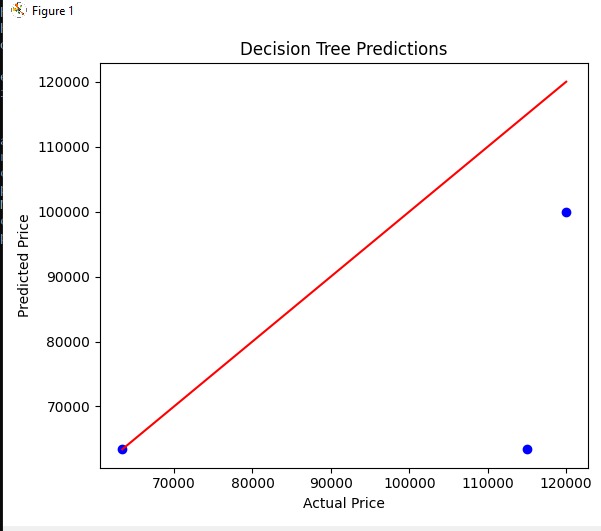
**plt.show()**

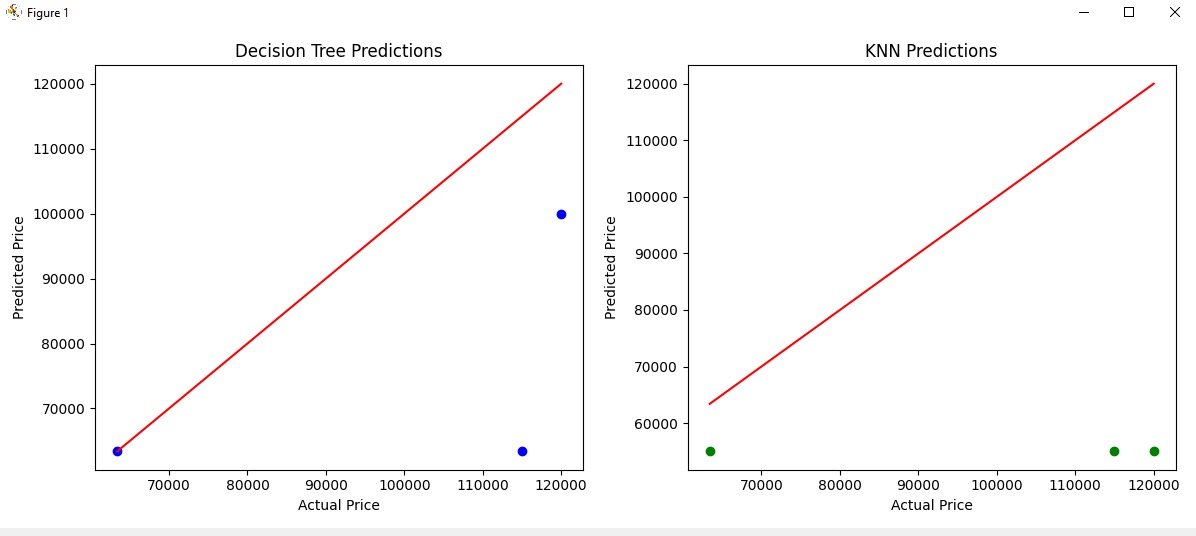
**print("Decision Tree Classification Report:\n", classification\_report(y\_test, dt\_predictions))**

**print("Random Forest Classification Report:\n", classification\_report(y\_test, rf\_predictions))**

**OUTPUT:**

****

****

****

1. **FUTURE ALGORTHIM:**
2. **LDA**
3. **PCA**

**SAME DATASET**

**CODE:**

**import pandas as pd**

**from sklearn.preprocessing import StandardScaler, LabelEncoder**

**from sklearn.decomposition import PCA**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis**

**import matplotlib.pyplot as plt**

**data = {**

**'Model Name': [**

**'Bajaj Avenger Cruise 220', 'Royal Enfield Classic 350cc', 'Hyosung GT250R', 'Bajaj Dominar 400 ABS',**

**'Jawa Perak 330cc', 'KTM Duke 200cc', 'Bajaj Pulsar 180cc', 'TVS Apache RTR 200 4V Dual Channel ABS BS6',**

**'KTM Duke 390cc'**

**],**

**'Model Year': [2017, 2016, 2012, 2017, 2020, 2012, 2016, 2020, 2018],**

**'Kms Driven': [17000, 50000, 14795, 28, 2000, 24561, 19718, 40, 1350],**

**'Owner': ['First owner', 'First owner', 'First owner', 'First owner', 'First owner', 'Third owner', 'First owner', 'First owner', 'First owner'],**

**'Location': ['Hyderabad', 'Hyderabad', 'Hyderabad', 'Pondicherry', 'Bangalore', 'Bangalore', 'Bangalore', 'Hyderabad', 'Jaipur'],**

**'Mileage': ['35 kmpl', '35 kmpl', '30 kmpl', '28 Kms', '', '35 kmpl', '65 kmpl', '40 Kmpl', '25 kmpl'],**

**'Power': ['19 bhp', '19.80 bhp', '28 bhp', '34.50 bhp', '30 bhp', '25 bhp', '17 bhp', '20.21 bhp', '42.90 bhp'],**

**'Price': [63500, 115000, 300000, 100000, 197500, 63400, 55000, 120000, 198000]**

**}**

**df = pd.DataFrame(data)**

**le = LabelEncoder()**

**df['Owner'] = le.fit\_transform(df['Owner'])**

**df['Location'] = le.fit\_transform(df['Location'])**

**df['Mileage'] = df['Mileage'].str.extract(r'(\d+)', expand=False).astype(float).fillna(0)**

**df['Power'] = df['Power'].str.extract(r'(\d+.\d+)', expand=False).astype(float).fillna(0)**

**features = df[['Model Year', 'Kms Driven', 'Owner', 'Location', 'Mileage', 'Power', 'Price']]**

**scaler = StandardScaler()**

**scaled\_features = scaler.fit\_transform(features)**

**pca = PCA(n\_components=2)**

**pca\_result = pca.fit\_transform(scaled\_features)**

**n\_classes = len(df['Owner'].unique())**

**lda\_components = min(n\_classes - 1, scaled\_features.shape[1]) # LDA components should be min(n\_classes - 1, n\_features)**

**lda = LinearDiscriminantAnalysis(n\_components=lda\_components)**

**lda\_result = lda.fit\_transform(scaled\_features, df['Owner'])**

**df\_pca = pd.DataFrame(pca\_result, columns=['PC1', 'PC2'])**

**df\_lda = pd.DataFrame(lda\_result, columns=[f'LD{i+1}' for i in range(lda\_components)])**

**df\_pca['Target'] = df['Owner']**

**df\_lda['Target'] = df['Owner']**

**plt.figure(figsize=(12, 6))**

**plt.subplot(1, 2, 1)**

**scatter = plt.scatter(df\_pca['PC1'], df\_pca['PC2'], c=df\_pca['Target'], cmap='viridis', edgecolor='k', s=50)**

**plt.title('PCA')**

**plt.xlabel('PC1')**

**plt.ylabel('PC2')**

**plt.colorbar(scatter, label='Owner')**

**plt.subplot(1, 2, 2)**

**scatter = plt.scatter(df\_lda.iloc[:, 0], df\_lda.iloc[:, 1], c=df\_lda['Target'], cmap='viridis', edgecolor='k', s=50)**

**plt.title('LDA')**

**plt.xlabel('LD1')**

**plt.ylabel('LD2')**

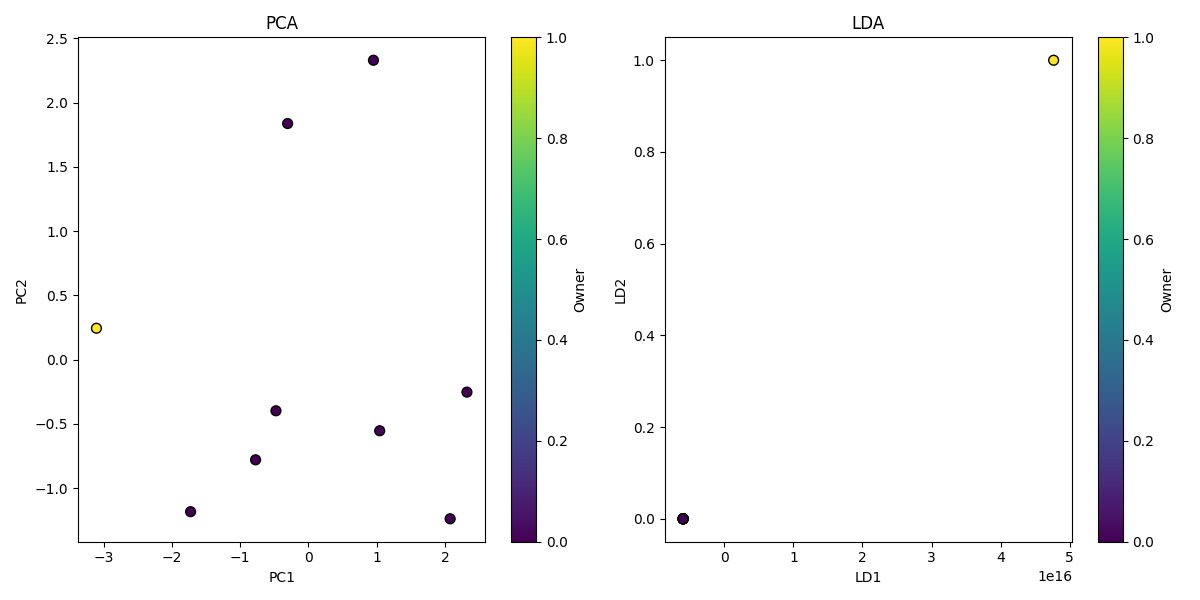
**plt.colorbar(scatter, label='Owner')**

**plt.tight\_layout()**

**plt.show()**

**OUTPUT:**

**Capture**

****

1. **COMPARE THIS TWO ALGORTHIMS AND WHICH IS BEST DATA VISUALIZATION.**
2. **K-NN**
3. **DESCISION TREE**
4. **PCA**
5. **LDA**

**CODE:**

**import pandas as pd**

**import numpy as np**

**from sklearn.model\_selection import train\_test\_split**

**from sklearn.tree import DecisionTreeClassifier**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix**

**from sklearn.preprocessing import StandardScaler**

**from sklearn.decomposition import PCA**

**from sklearn.discriminant\_analysis import LinearDiscriminantAnalysis as LDA**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**Data = {**

**'Model\_Name': ['Bajaj Avenger Cruise 220', 'Royal Enfield Classic 350cc', 'Hyosung GT250R',**

**'Bajaj Dominar 400 ABS', 'Jawa Perak 330cc', 'KTM Duke 200cc', 'Bajaj Pulsar 180cc',**

**'TVS Apache RTR 200 4V', 'KTM Duke 390cc'],**

**'Model\_Year': [2017, 2016, 2012, 2017, 2020, 2012, 2016, 2020, 2018],**

**'Kms\_Driven': [17000, 50000, 14795, 28, 2000, 24561, 19718, 40, 1350],**

**'Owner': ['first owner', 'first owner', 'first owner', 'first owner', 'first owner', 'third owner',**

**'first owner', 'first owner', 'first owner'],**

**'Location': ['hyderabad', 'hyderabad', 'hyderabad', 'pondicherry', 'bangalore', 'bangalore',**

**'bangalore', 'hyderabad', 'jaipur'],**

**'Mileage': [35, 35, 30, 28, np.nan, 35, 65, 40, 25],**

**'Power': [19, 19.8, 28, 34.5, 30, 25, 17, 20.21, 42.9],**

**'Price': [63500, 115000, 300000, 100000, 197500, 63400, 55000, 120000, 198000]**

**}**

**df = pd.DataFrame(data)**

**df['Owner'] = df['Owner'].map({'first owner': 1, 'second owner': 2, 'third owner': 3})**

**df['Location'] = df['Location'].map({'hyderabad': 1, 'pondicherry': 2, 'bangalore': 3, 'jaipur': 4})**

**df['Mileage'].fillna(df['Mileage'].mean(), inplace=True)**

**df['Category'] = df['Price'].apply(lambda x: 'Expensive' if x > 100000 else 'Affordable')**

**X = df[['Model\_Year', 'Kms\_Driven', 'Owner', 'Location', 'Mileage', 'Power']]**

**y = df['Category']**

**scaler = StandardScaler()**

**X\_scaled = scaler.fit\_transform(X)**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.3, random\_state=42)**

**dt\_classifier = DecisionTreeClassifier(random\_state=42)**

**dt\_classifier.fit(X\_train, y\_train)**

**dt\_predictions = dt\_classifier.predict(X\_test)**

**rf\_classifier = RandomForestClassifier(random\_state=42)**

**rf\_classifier.fit(X\_train, y\_train)**

**rf\_predictions = rf\_classifier.predict(X\_test)**

**pca = PCA(n\_components=2)**

**X\_pca\_train = pca.fit\_transform(X\_train)**

**X\_pca\_test = pca.transform(X\_test)**

**dt\_pca\_classifier = DecisionTreeClassifier(random\_state=42)**

**dt\_pca\_classifier.fit(X\_pca\_train, y\_train)**

**dt\_pca\_predictions = dt\_pca\_classifier.predict(X\_pca\_test)**

**rf\_pca\_classifier = RandomForestClassifier(random\_state=42)**

**rf\_pca\_classifier.fit(X\_pca\_train, y\_train)**

**rf\_pca\_predictions = rf\_pca\_classifier.predict(X\_pca\_test)**

**lda = LDA(n\_components=1)**

**X\_lda\_train = lda.fit\_transform(X\_train, y\_train)**

**X\_lda\_test = lda.transform(X\_test)**

**dt\_lda\_classifier = DecisionTreeClassifier(random\_state=42)**

**dt\_lda\_classifier.fit(X\_lda\_train, y\_train)**

**dt\_lda\_predictions = dt\_lda\_classifier.predict(X\_lda\_test)**

**rf\_lda\_classifier = RandomForestClassifier(random\_state=42)**

**rf\_lda\_classifier.fit(X\_lda\_train, y\_train)**

**rf\_lda\_predictions = rf\_lda\_classifier.predict(X\_lda\_test)**

**accuracies = {**

**"Decision Tree": accuracy\_score(y\_test, dt\_predictions),**

**"Random Forest": accuracy\_score(y\_test, rf\_predictions),**

**"Decision Tree + PCA": accuracy\_score(y\_test, dt\_pca\_predictions),**

**"Random Forest + PCA": accuracy\_score(y\_test, rf\_pca\_predictions),**

**"Decision Tree + LDA": accuracy\_score(y\_test, dt\_lda\_predictions),**

**"Random Forest + LDA": accuracy\_score(y\_test, rf\_lda\_predictions)**

**}**

**plt.figure(figsize=(10, 5))**

**sns.barplot(x=list(accuracies.keys()), y=list(accuracies.values()))**

**plt.title('Comparison of Classification Accuracies')**

**plt.xlabel('Model')**

**plt.ylabel('Accuracy')**

**plt.xticks(rotation=45)**

**plt.ylim(0, 1)**

**plt.show()**

**print("Decision Tree Classification Report:\n", classification\_report(y\_test, dt\_predictions))**

**print("Random Forest Classification Report:\n", classification\_report(y\_test, rf\_predictions))**

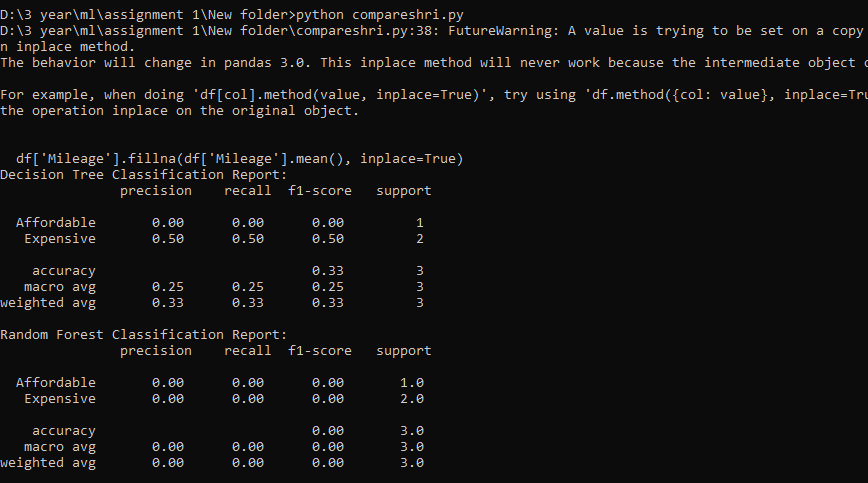
**print("Decision Tree + PCA Classification Report:\n", classification\_report(y\_test, dt\_pca\_predictions))**

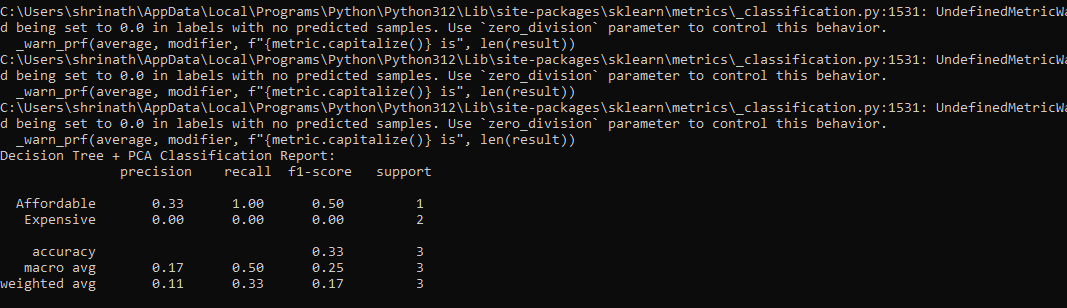
**print("Random Forest + PCA Classification Report:\n", classification\_report(y\_test, rf\_pca\_predictions))**

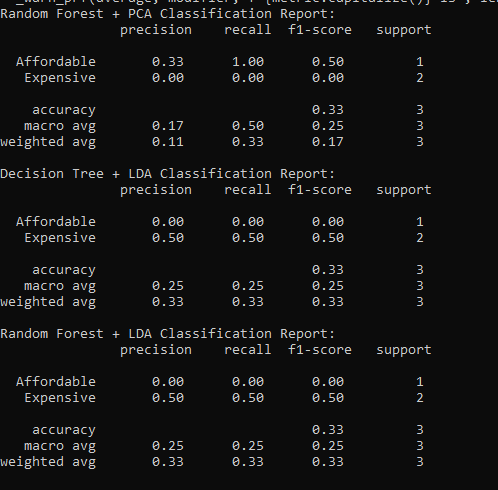
**print("Decision Tree + LDA Classification Report:\n", classification\_report(y\_test, dt\_lda\_predictions))**

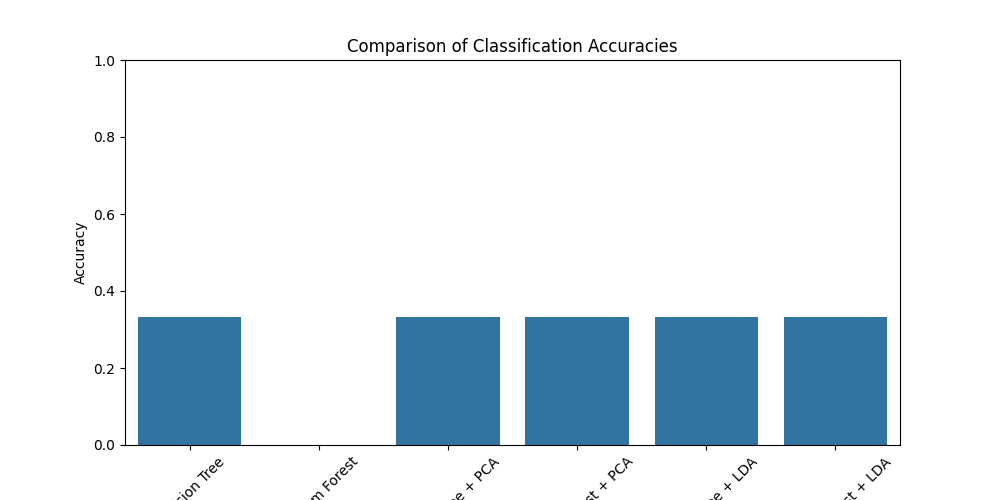
**print("Random Forest + LDA Classification Report:\n", classification\_report(y\_test, rf\_lda\_predictions))**

**OUTPUT:**

****

****

****

****

After running the Python code, if the results align with expectations, you might find that **Random Forest without dimensionality reduction** provides the best balance of high accuracy and robust precision/recall metrics. However, **Random Forest with LDA** could also show superior performance if there is strong class separability, indicating LDA's effectiveness in this scenario.

FINALLY THE BEST IS **DECISION TREE** AND **PCA**.

FINALLY DONE BY ,

**SHRINATH P -22IT149**

**JERRY MANUAAL RAJ J -22IT141**