Understood. Here's a brief explanation of the codes in the entire project, including the exploratory data analysis (EDA) with the new instance:

1. Importing Libraries: The initial step involves importing necessary libraries for data handling, modelling, evaluation, and visualization.

from sklearn.datasets import load\_wine

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

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

2. Loading the Dataset: The Wine dataset is loaded using `load\_wine()` from scikit-learn, providing access to its features and target labels.

wine\_dataset = load\_wine()

3. Data Preprocessing: Missing values are checked and handled if necessary. The dataset is split into features (X) and target labels (y), and preprocessing steps such as scaling are applied using `StandardScaler`.

X = wine\_dataset.data

y = wine\_dataset.target

scaler = StandardScaler()

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

4. Train-Test Split: The dataset is split into training and testing sets using `train\_test\_split()` with stratified sampling to maintain class balance.

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42, stratify=y)

5. Model Training - Logistic Regression: A logistic regression model is instantiated and trained on the training data using `Logistic Regression`.

logistic\_model = LogisticRegression(max\_iter=10000)

logistic\_model.fit(X\_train, y\_train)

6. Model Training - Decision Trees: A decision tree classifier is instantiated and trained on the training data using `DecisionTreeClassifier`.

decision\_tree\_model = DecisionTreeClassifier(random\_state=42)

decision\_tree\_model.fit(X\_train, y\_train)

7. Model Training - SVM: A support vector machine (SVM) classifier with an RBF kernel is instantiated and trained on the training data using `SVC`.

svm\_model = SVC(kernel='rbf', random\_state=42)

svm\_model.fit(X\_train, y\_train)

8. Model Evaluation: The performance of each model is evaluated using metrics such as accuracy, precision, recall, and F1-score, calculated using scikit-learns metrics functions.

y\_pred\_logistic = logistic\_model.predict(X\_test)

accuracy\_lr = accuracy\_score(y\_test, y\_pred\_logistic)

precision\_lr = precision\_score(y\_test, y\_pred\_logistic, average='weighted')

recall\_lr = recall\_score(y\_test, y\_pred\_logistic, average='weighted')

f1\_lr = f1\_score(y\_test, y\_pred\_logistic, average='weighted')

9. Exploratory Data Analysis (EDA): Summary statistics, pair plots, and boxplots are generated to understand the dataset's characteristics and distributions. Additionally, correlation heatmaps are created to visualize feature interactions.

# Summary statistics

print("Summary Statistics:")

print(df.describe())

# Pairplot

plt.figure(figsize=(12, 8))

sns.pairplot(df, hue='target', palette='viridis')

plt.title("Pairplot of Wine Dataset")

plt.show()

# Correlation heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap of Wine Dataset")

plt.show()

10. EDA with New Instance: Similar EDA techniques are applied to explore the characteristics of the new instances, aiding in assessing their suitability for classification tasks.

# Summary statistics of the new instances

print("Summary Statistics of New Instances:")

print(new\_instances\_df.describe())

# Boxplot of features in the new instances

plt.figure(figsize=(12, 8))

sns.boxplot(data=new\_instances\_df, palette='viridis')

plt.title("Boxplot of Features in New Instances")

plt.xticks(rotation=45)

plt.show()

# Correlation heatmap of features in the new instances

plt.figure(figsize=(10, 8))

sns.heatmap(new\_instances\_df.corr(), annot=True, cmap='coolwarm')

plt.title("Correlation Heatmap of Features in New Instances")

plt.show()

11. Results Analysis: The results for each model are analysed and discussed, including interpretation of coefficients for logistic regression (if applicable) and insights into model performance.

conf\_matrix\_lr = confusion\_matrix(y\_test, y\_pred\_logistic)

plt.figure(figsize=(8, 6))

sns.heatmap(conf\_matrix\_lr, annot=True, fmt="d", cmap="Blues", xticklabels=wine\_dataset.target\_names, yticklabels=wine\_dataset.target\_names)

plt.title("Confusion Matrix - Logistic Regression")

plt.xlabel("Predicted Label")

plt.ylabel("True Label")

plt.show()

These codes collectively form the project's workflow, encompassing data preprocessing, model training, evaluation, analysis, and documentation.