```
from google.colab import files
uploaded = files.upload()
```

Choose files IRIS.csv

- **IRIS.csv**(text/csv) - 4617 bytes, last modified: 11/08/2023 - 100% done

```
Saving IRIS.csv to IRIS.csv
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, ConfusionMatrixDisplay, roc_curve, auc

# Load the dataset
df = pd.read_csv("IRIS.csv")

# Display basic information about the dataset
print("First few rows of the dataset:")
print(df.head())
print("\nInformation about the dataset:")
print(df.info())
print("\nSummary statistics of the dataset:")
print(df.describe())
print("\nCounts of each species:")
print(df['species'].value_counts())

# Visualize data using plots
sns.countplot(data=df, x='species')
plt.show()

sns.scatterplot(data=df, x='petal_length', y='petal_width', hue='species')
plt.show()

sns.scatterplot(data=df, x='sepal_length', y='sepal_width', hue='species')
plt.show()

sns.pairplot(data=df, hue='species')
plt.show()

# Prepare the data for modeling
X = df.drop('species', axis=1)
y = df['species']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=101)
scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)

# Create a logistic regression model and perform grid search for hyperparameters
log_model = LogisticRegression(solver='saga', multi_class='ovr', max_iter=5000)
param_grid = {'penalty': ['l1', 'l2'], 'C': np.logspace(0, 10, 10)}
grid_model = GridSearchCV(log_model, param_grid=param_grid)
grid_model.fit(scaled_X_train, y_train)

# Evaluate the model's performance
y_pred = grid_model.predict(scaled_X_test)

# Display accuracy, confusion matrix, and classification report
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))

# Display a confusion matrix using a heatmap
ConfusionMatrixDisplay.from_estimator(grid_model, scaled_X_test, y_test)

# Define a function to plot ROC curves for multi-class classification
def plot_multiclass_roc(clf, X_test, y_test, n_classes, figsize=(5, 5)):
    # Calculate ROC curve values
    y_score = clf.decision_function(X_test)
    fpr = dict()
    tpr = dict()
    roc_auc = dict()

    # Calculate dummies for each class
    y_test_dummies = pd.get_dummies(y_test, drop_first=False).values
    for i in range(n_classes):
        fpr[i], tpr[i], _ = roc_curve(y_test_dummies[:, i], y_score[:, i])
```

```
            fpr[i], tpr[i], _ = roc_curve(y_test_dummies[:, i], y_score[:, i])
            roc_auc[i] = auc(fpr[i], tpr[i])

        # Plot ROC curves
        fig, ax = plt.subplots(figsize=figsize)
        ax.plot([0, 1], [0, 1], 'k--')
        ax.set_xlim([0.0, 1.0])
        ax.set_ylim([0.0, 1.05])
        ax.set_xlabel('False Positive Rate')
        ax.set_ylabel('True Positive Rate')
        ax.set_title('Receiver Operating Characteristic')
        for i in range(n_classes):
            ax.plot(fpr[i], tpr[i], label=f'Class {i} (AUC = {roc_auc[i]:.2f})')
        ax.legend(loc="best")
        ax.grid(alpha=0.4)
        sns.despine()
        plt.show()

# Plot ROC curves for the multi-class model
plot_multiclass_roc(grid_model, scaled_X_test, y_test, n_classes=3, figsize=(16, 10))
```

```
First few rows of the dataset:
   sepal_length  sepal_width  petal_length  petal_width      species
0           5.1          3.5           1.4          0.2  Iris-setosa
1           4.9          3.0           1.4          0.2  Iris-setosa
2           4.7          3.2           1.3          0.2  Iris-setosa
3           4.6          3.1           1.5          0.2  Iris-setosa
4           5.0          3.6           1.4          0.2  Iris-setosa

Information about the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
None

Summary statistics of the dataset:
       sepal_length  sepal_width  petal_length  petal_width
count    150.000000   150.000000    150.000000   150.000000
mean       5.843333     3.054000      3.758667     1.198667
std        0.828066     0.433594      1.764420     0.763161
min        4.300000     2.000000      1.000000     0.100000
25%        5.100000     2.800000      1.600000     0.300000
50%        5.800000     3.000000      4.350000     1.300000
75%        6.400000     3.300000      5.100000     1.800000
max        7.900000     4.400000      6.900000     2.500000

Counts of each species:
Iris-setosa        50
Iris-versicolor    50
Iris-virginica     50
Name: species, dtype: int64
```