```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_absolute_error, mean_squared_error,
r2_score
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR
import numpy as np

from google.colab import files

uploaded = files.upload()
file_name = list(uploaded.keys())[0]

try:
    data = pd.read_csv(file_name)
except pd.errors.ParserError:
    print("Error parsing the CSV file. Please check the file format and
delimiters.")
except Exception as e:
    print(f"An error occurred: {e}")

print(data.head())
print(data.info())
print(data.describe())

sns.countplot(data=data, x='Delay')
plt.title('Delay Distribution')
plt.show()

sns.scatterplot(data=data, x='Length', y='Delay')
plt.title('Flight Length vs Delay')
plt.show()


print(data.isnull().sum())
data = data.dropna()


X = data.iloc[:, 1:-1]
y = data.iloc[:, -1]
```

```
le_airline = LabelEncoder()
X['Airline'] = le_airline.fit_transform(X['Airline'])
X['AirportFrom'] = le_airline.fit_transform(X['AirportFrom'])
X['AirportTo'] = le_airline.fit_transform(X['AirportTo'])


X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.3, random_state=42)

from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR

gbr = GradientBoostingRegressor(n_estimators=100, learning_rate=0.1,
max_depth=3, random_state=42)
gbr.fit(X_train, y_train)

svr = SVR(kernel='rbf', C=100, gamma='auto')
svr.fit(X_train, y_train)

y_pred_gbr = gbr.predict(X_test)
print("Gradient Boosting Regressor Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_gbr))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_gbr))
print("R-squared:", r2_score(y_test, y_pred_gbr))

y_pred_svr = svr.predict(X_test)
print("\nSupport Vector Regressor Performance:")
print("Mean Absolute Error:", mean_absolute_error(y_test, y_pred_svr))
print("Mean Squared Error:", mean_squared_error(y_test, y_pred_svr))
print("R-squared:", r2_score(y_test, y_pred_svr))

if hasattr(gbr, 'feature_importances_'):
    importances = gbr.feature_importances_
    features = X.columns
    indices = np.argsort(importances)

    plt.figure()
    plt.title('Feature Importance (Gradient Boosting)')
    plt.barh(range(X.shape[1]), importances[indices], align='center')
    plt.yticks(range(X.shape[1]), [features[i] for i in indices])
    plt.xlabel('Feature Importance')
```

```
    plt.show()
def predict_new_data(model, input_data):
    prediction = model.predict(input_data)
    return prediction

new_data = pd.DataFrame({
     'Airline': [0],
     'Flight': [123],
     'AirportFrom': [1],
     'AirportTo': [2],
     'DayOfWeek': [3],
     'Time': [15],
     'Length': [200]
 })
print("Prediction:", predict_new_data(gbr, new_data))
```