AIM

To develop and compare predictive models for flight arrival delay utilizing Stochastic Gradient Descent (SGD), Linear Regression, and Logistic Regression techniques.

ALGORITHM

- Import the neccesary packages.
- Load the dataset using pd.read_csv.
- Obtain summary statistics with describe.
- Display column names using columns.
- Plot 'DISTANCE' for the first 200 entries using plt.plot.
- Utilize count plots (sns.countplot) for 'AIRLINE_CODE' and 'DOT_CODE'.
- Generate histograms for the first five numeric columns using sns.histplot.
- Convert 'FL_DATE' to datetime using pd.to_datetime.
- Drop rows with missing values in 'ARR_DELAY'.
- Select features and target variable.
- Split the data into training and testing sets using train_test_split.
- Standardize features using StandardScaler.
- Train an SGDRegressor model and make predictions.
- Evaluate the model using Mean Squared Error and R2 score.
- Create a pair plot using sns.pairplot to visualize relationships between selected
  columns with 'ARR_DELAY' as the hue.
- Train a Linear Regression model using LinearRegression.
- Predict arrival delays on the test set.
- Evaluate the model's performance using R2 score.
- Train a Logistic Regression model using LogisticRegression.
- Predict arrival delays on the test set.
- Evaluate the model's performance using R2 score.
- Atlast compare R2 score of SGD,linear,logistic regression and come to conclusion

CODE

In [ ]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV

from sklearn.ensemble import RandomForestRegressor

from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import SGDClassifier
from sklearn.linear_model import SGDRegressor

from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
```

```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
```

In [ ]:
```python
Dataset=pd.read_csv("/content/drive/MyDrive/2019.csv")
```

In [ ]:
```python
Dataset.describe()
```

Out[ ]:

	DOT_CODE	FL_NUMBER	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	WHEE
count	7.268232e+06	7.268232e+06	7.268232e+06	7.268232e+06	7.268232e+06	7.268232e+06	7.2682
mean	1.998619e+04	2.548655e+03	1.329149e+03	1.334412e+03	1.084572e+01	1.738121e+01	1.3581
std	3.739468e+02	1.796975e+03	4.928371e+02	5.072381e+02	4.878693e+01	9.991259e+00	5.0886
min	1.939300e+04	1.000000e+00	1.000000e+00	1.000000e+00	-8.200000e+01	1.000000e+00	1.0000
25%	1.979000e+04	1.018000e+03	9.120000e+02	9.140000e+02	-5.000000e+00	1.100000e+01	9.3000
50%	1.997700e+04	2.149000e+03	1.320000e+03	1.327000e+03	-2.000000e+00	1.500000e+01	1.3400
75%	2.036800e+04	3.900000e+03	1.735000e+03	1.746000e+03	7.000000e+00	2.000000e+01	1.8010
max	2.045200e+04	7.933000e+03	2.359000e+03	2.400000e+03	2.710000e+03	2.270000e+02	2.4000

8 rows × 26 columns

In [ ]:
```python
Dataset.shape
```

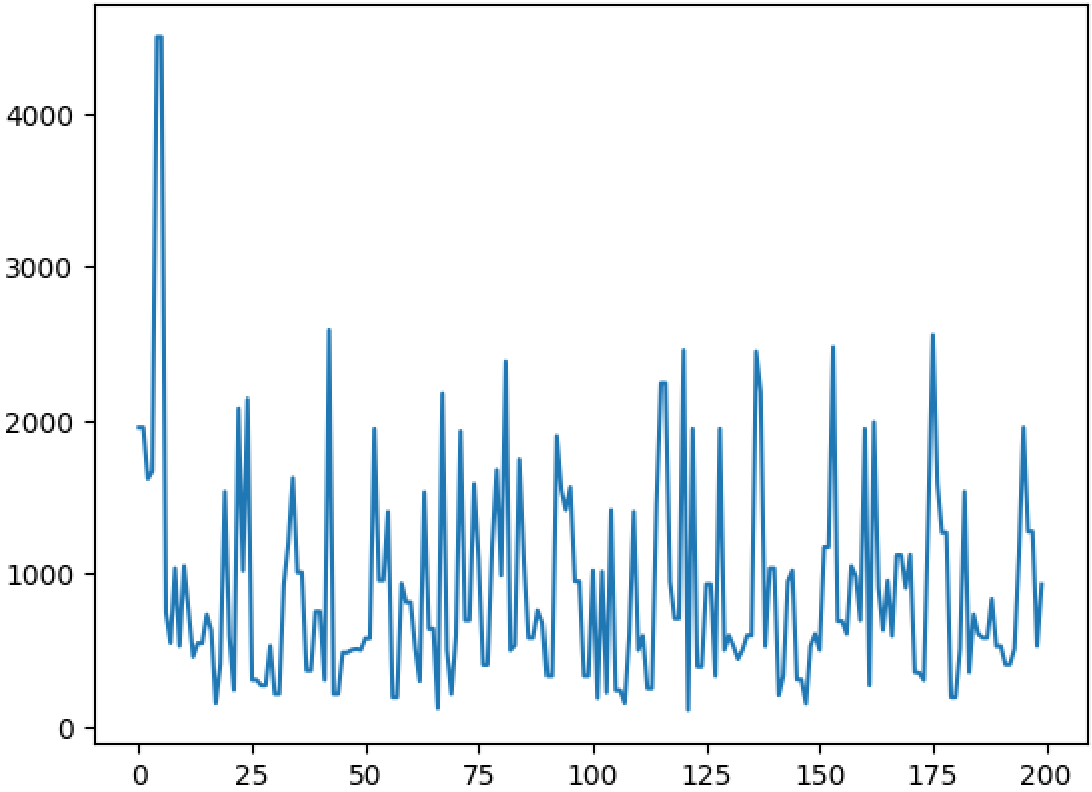
Out[ ]:
```
(7268232, 33)
```

In [ ]:
```python
Dataset.columns
```

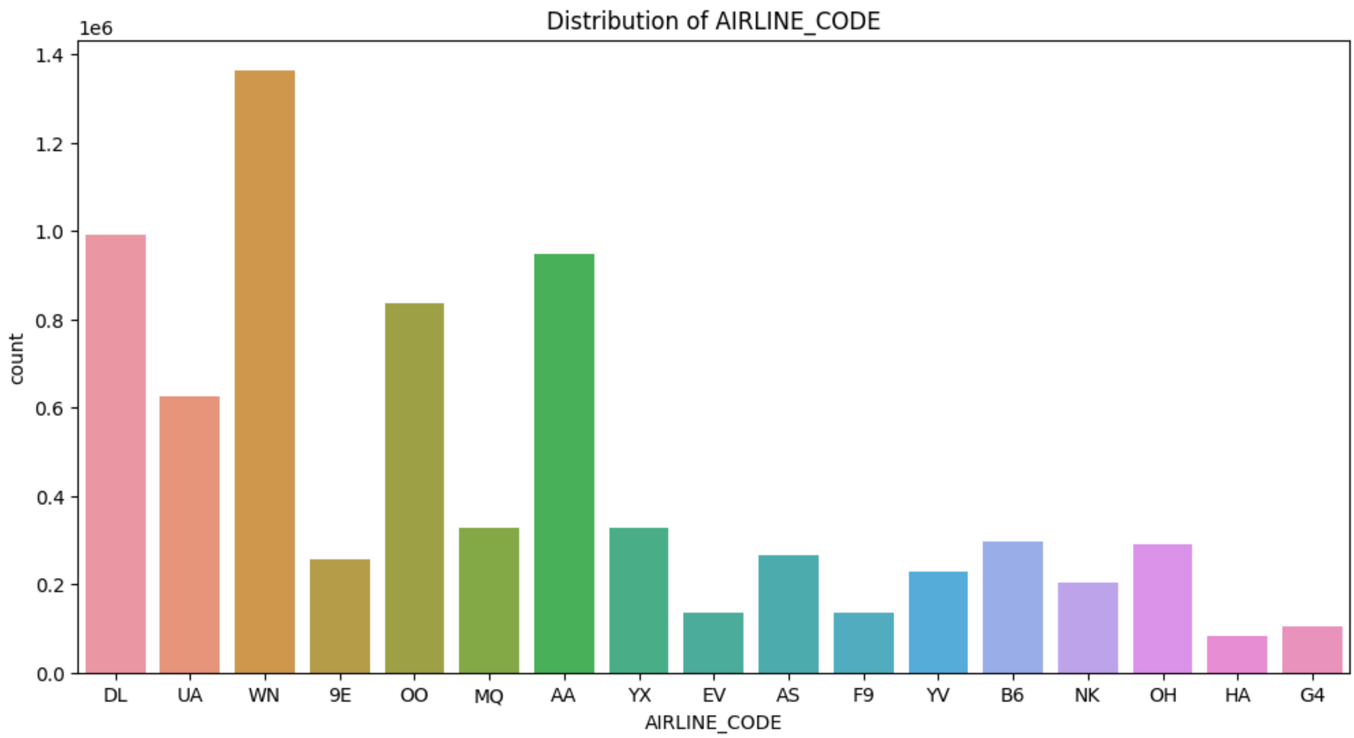
Out[ ]:
```
Index(['FL_DATE', 'AIRLINE_CODE', 'DOT_CODE', 'FL_NUMBER', 'ORIGIN',
       'ORIGIN_CITY', 'DEST', 'DEST_CITY', 'CRS_DEP_TIME', 'DEP_TIME',
       'DEP_DELAY', 'TAXI_OUT', 'WHEELS_OFF', 'WHEELS_ON', 'TAXI_IN',
       'CRS_ARR_TIME', 'ARR_TIME', 'ARR_DELAY', 'CANCELLED',
       'CANCELLATION_CODE', 'DIVERTED', 'CRS_ELAPSED_TIME', 'ELAPSED_TIME',
       'AIR_TIME', 'DISTANCE', 'DELAY_DUE_CARRIER', 'DELAY_DUE_WEATHER',
       'DELAY_DUE_NAS', 'DELAY_DUE_SECURITY', 'DELAY_DUE_LATE_AIRCRAFT',
       'FL_YEAR', 'FL_MONTH', 'FL_DAY'],
      dtype='object')
```

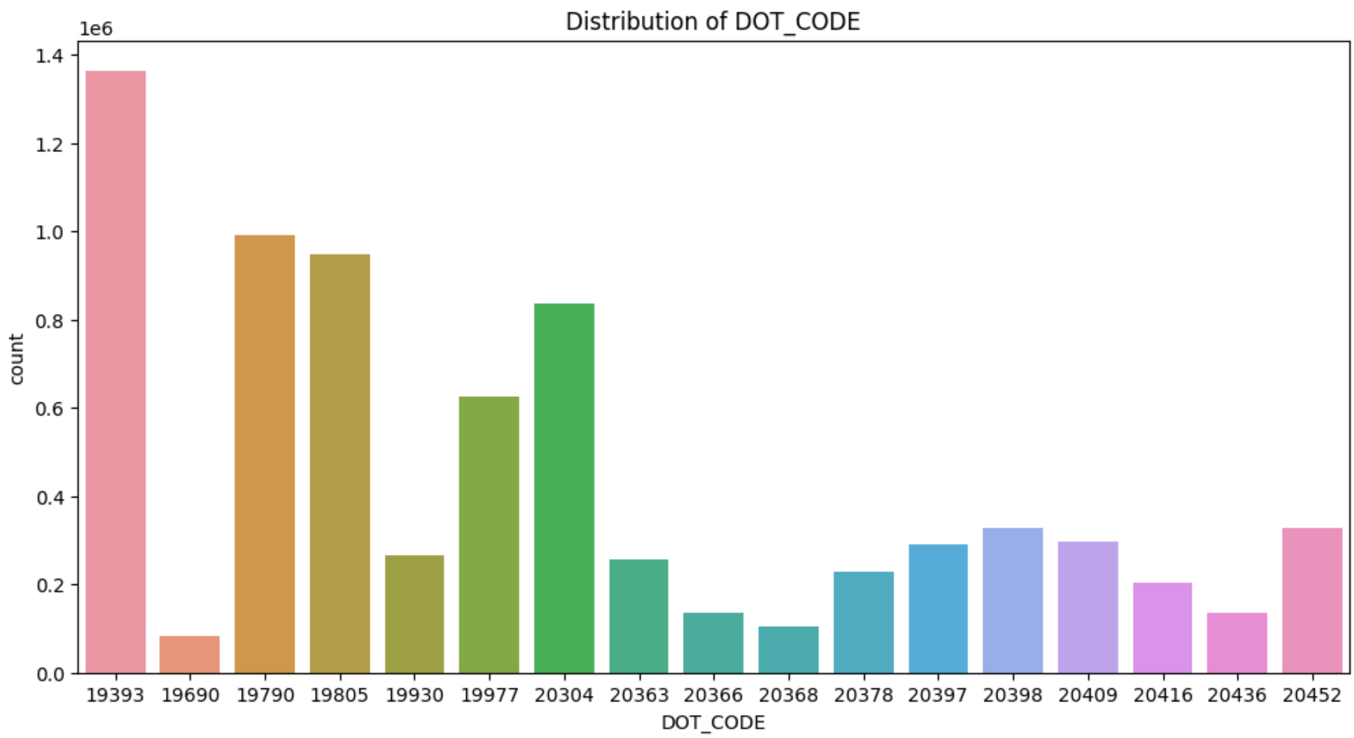
In [ ]:
```python
Dataset['FL_DATE']=pd.to_datetime(Dataset['FL_DATE'])
plt.plot(Dataset.index[:200],Dataset['DISTANCE'].head(200))
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x7aa4c4d0cf40>]
```



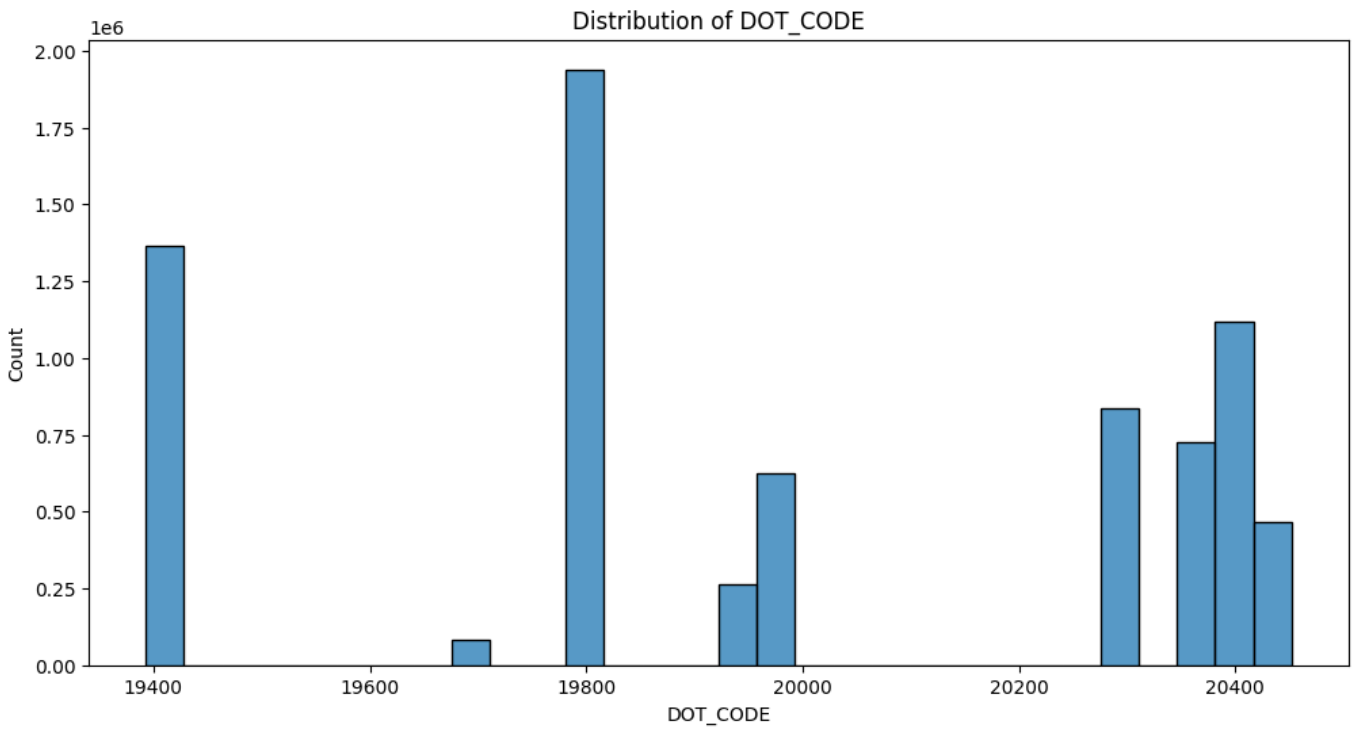
In [ ]:
```
#Explore AIRLINE_CODE categorical column
plt.figure(figsize=(12, 6))
sns.countplot(x='AIRLINE_CODE', data=Dataset)
plt.title(f'Distribution of AIRLINE_CODE')
plt.show()
#Explore DOTCODE categorical column
plt.figure(figsize=(12, 6))
sns.countplot(x='DOT_CODE', data=Dataset)
plt.title(f'Distribution of DOT_CODE')
plt.show()
```

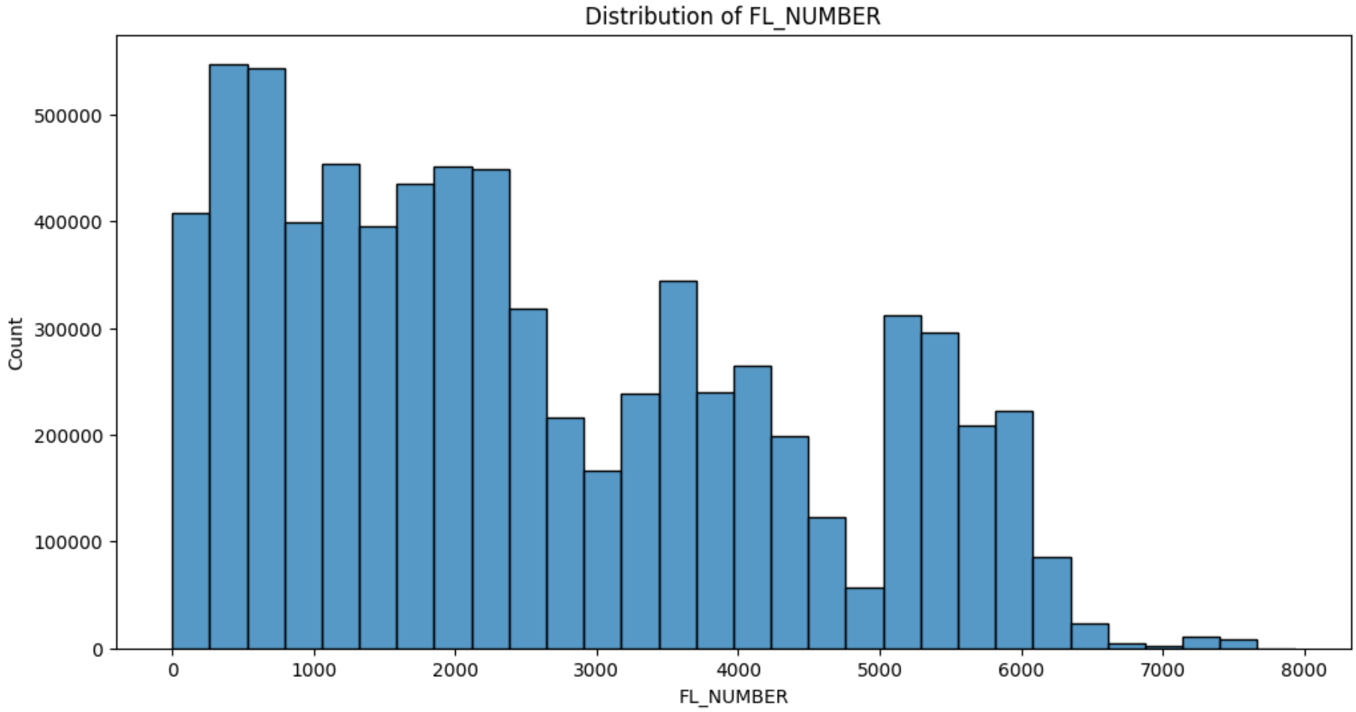


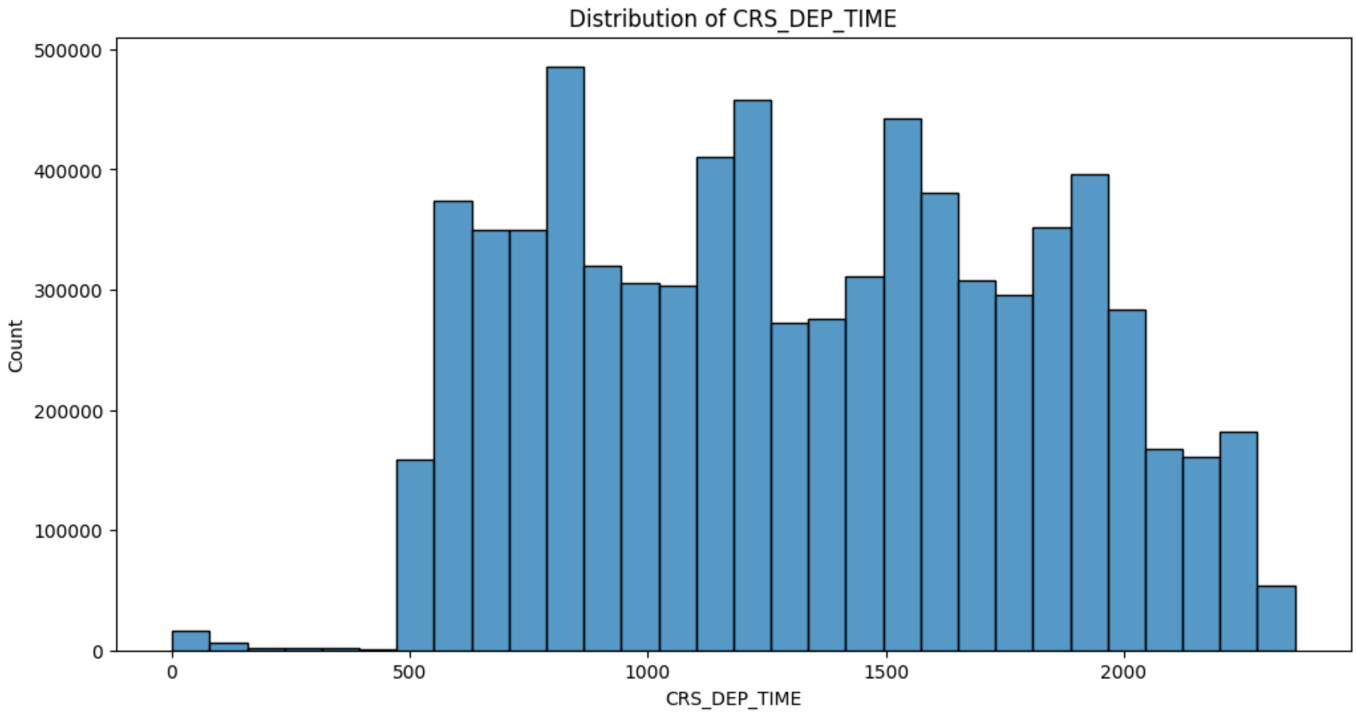


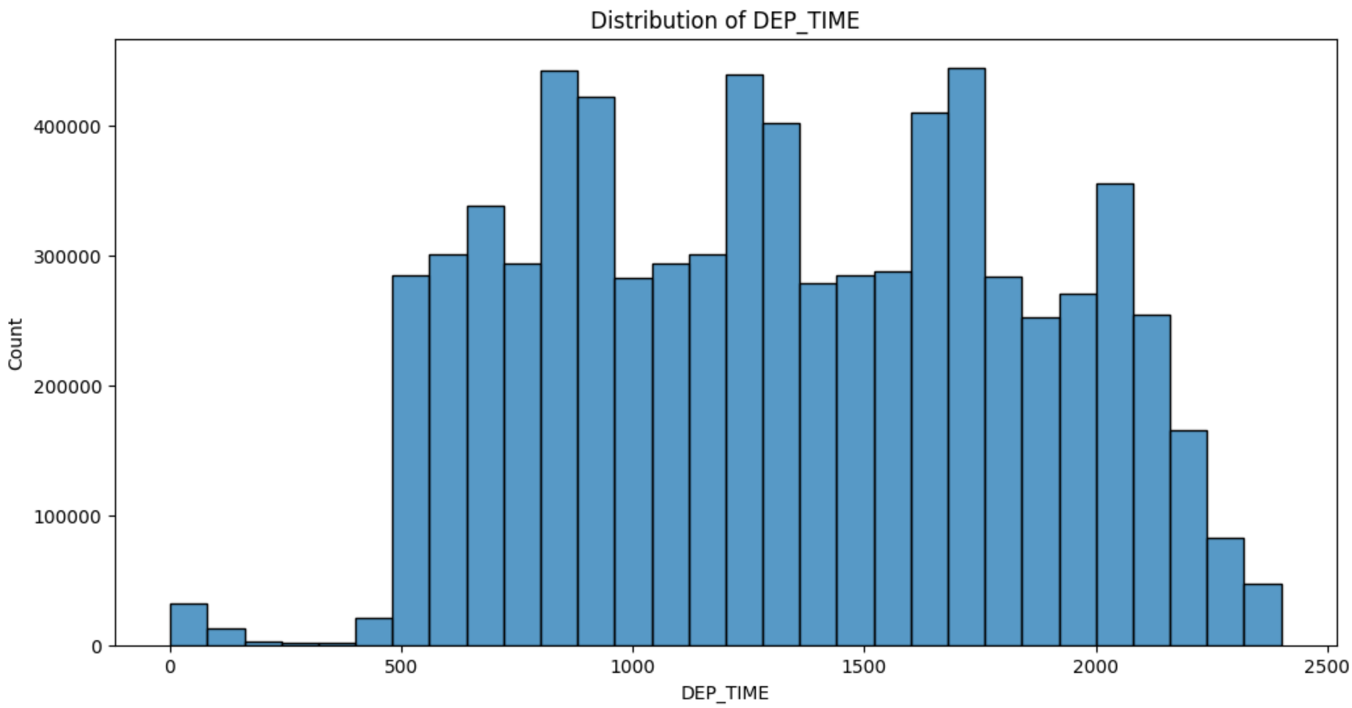
In [ ]:

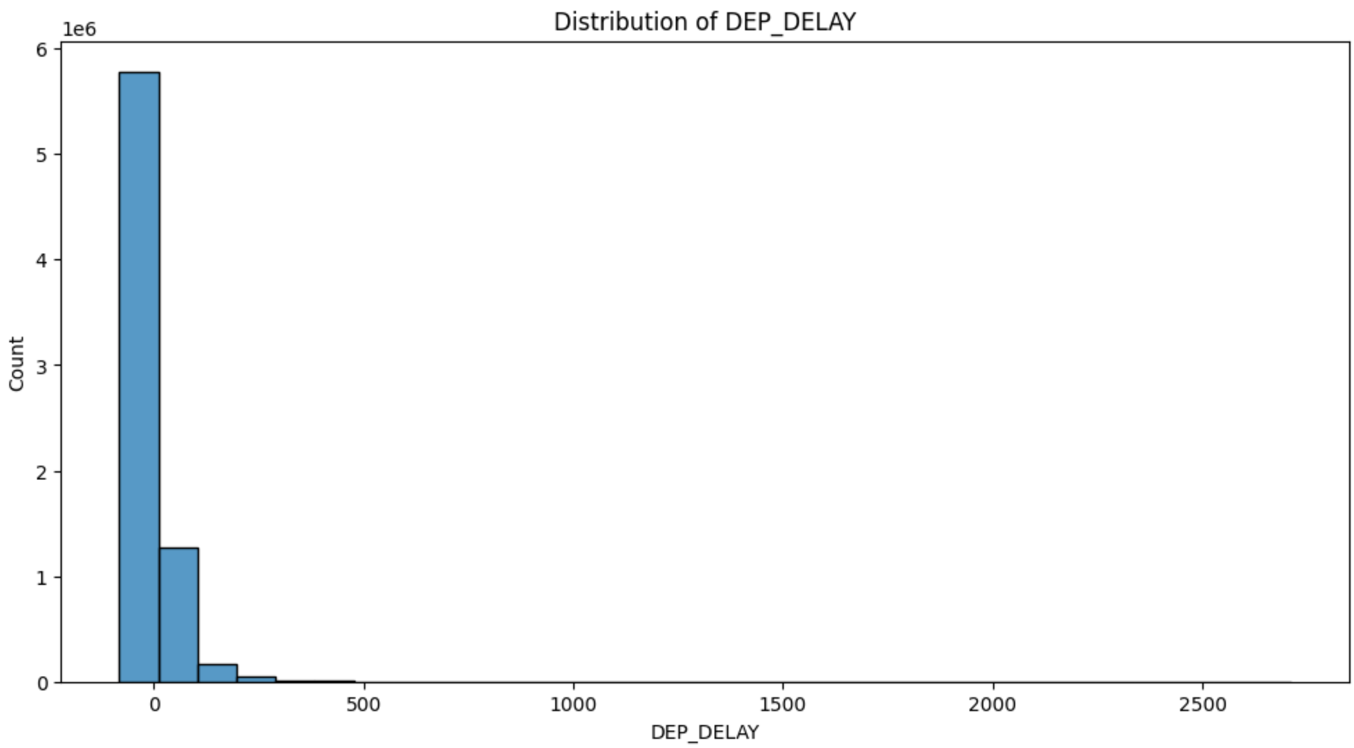
```python
# Explore numeric columns
numeric_cols = Dataset.select_dtypes(include=['number']).columns
for col in numeric_cols[:5]:
    plt.figure(figsize=(12, 6))
    sns.histplot(Dataset[col].dropna(), kde=False, bins=30)
    plt.title(f'Distribution of {col}')
    plt.show()
```




Distribution of FL_NUMBER
Count
0
100000
200000
300000
400000
500000
0
1000
2000
3000
4000
5000
6000
7000
8000
FL_NUMBER


Distribution of CRS_DEP_TIME
Count
0
100000
200000
300000
400000
500000
0
500
1000
1500
2000
CRS_DEP_TIME





In [ ]:

```
Dataset= Dataset.dropna(subset=['ARR_DELAY'])# Drop the rows with missing values
features = ['CRS_DEP_TIME', 'DEP_DELAY', 'TAXI_OUT', 'WHEELS_OFF', 'TAXI_IN', 'CRS_ARR_TIME']
target = 'ARR_DELAY'
X = Dataset[features]
y = Dataset[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)# Sp
scaler = StandardScaler()# Standardize features
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
sgd_regressor = SGDRegressor(max_iter=1000, tol=1e-3, random_state=42)# Initialize SGDRegress
sgd_regressor.fit(X_train_scaled, y_train)# Train the model
y_pred = sgd_regressor.predict(X_test_scaled)# Make predictions on the test set
print("Predicted Arrival delay",y_pred)
mse = mean_squared_error(y_test, y_pred)# Evaluate the model
print(f'Mean Squared Error: {mse}')
print('Coefficients:', sgd_regressor.coef_)# Print the coefficients and intercept
print('Intercept:', sgd_regressor.intercept_)
```

```
SGDR2=sgd_regressor.score(X_test_scaled, y_test)*100
print('Accuracy:', SGDR2)
```

```
Predicted Arrival delay [-13.04914207   1.9010553  -11.97859881 ... -10.51417566  -3.37723221
  79.4846428 ]
Mean Squared Error: 103.16070946017929
Coefficients: [-0.19622133 48.46043681  7.95988335  0.82278732  4.35438192 -0.41562734]
Intercept: [5.38537641]
Accuracy: 96.09798950935853
```

In [ ]:

```
print("Transformed data of X_Train using StandardScaler\n")
print(X_train_scaled,"\n\n")
print("Transformed data of X_Test using StandardScaler\n")
print(X_test_scaled)
```

```
Transformed data of X_Train using StandardScaler

[[-1.21606431 -0.36638022 -0.13860573 -1.21711709 -0.44259734 -0.82094307]
 [ 1.80732541 -0.32530978 -0.43890814  1.70899163 -0.60449121  1.5981571 ]
 [ 0.7826195  -0.32530978 -0.43890814  0.71659076  4.8999004   1.58280992]
 ...
 [-0.2217952   0.74252167  2.26381354 -0.02230574 -0.1188096  -0.25885239]
 [-1.36824835 -0.28423934  0.16169668 -1.2721413  -0.44259734 -1.01661969]
 [ 0.62028985 -0.2431689   1.96351113  0.69300895 -0.28070347  0.5200173 ]]


Transformed data of X_Test using StandardScaler

[[ 0.03996136 -0.30477456 -0.23870653  0.08184722 -0.44259734 -0.247342  ]
 [ 1.19858923 -0.36638022  0.06159588  1.12534201  3.11906782  1.18378309]
 [-1.06388026 -0.18156324 -0.53900894 -1.05990507 -0.92827895 -1.03580367]
 ...
 [ 0.35244593 -0.28423934  0.06159588  0.31373495 -0.60449121  0.41450539]
 [-0.21773696 -0.22263368  0.7623015  -0.21882076 -0.92827895 -0.33175152]
 [ 1.39135569  1.33804306  1.26280551  1.65593258 -0.28070347  1.59240191]]
```

In [ ]:

```
#LINEAR REGRESSION
Dataset= Dataset.dropna(subset=['ARR_DELAY'])
features = ['CRS_DEP_TIME', 'DEP_DELAY', 'TAXI_OUT', 'WHEELS_OFF', 'TAXI_IN', 'CRS_ARR_TIME']
target = 'ARR_DELAY'
X = Dataset[features]
y = Dataset[target]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
LR=LinearRegression()
LR.fit(X_train,y_train)
LRy_pred=LR.predict(X_test)
print("Y_predict value=",LRy_pred)
LR_R2=r2_score(LRy_pred,y_test)*100
print("R2 square = ",LR_R2)
```

```
Y_predict value= [-13.04884802   1.84570298 -11.81275597 ... -10.56481402  -3.38733674
  79.2474683 ]
R2 square =  95.93964893735287
```

In [ ]:

```
#LOGISTIC REGRESSION


Dataset= Dataset.dropna(subset=['ARR_DELAY'])
features = ['CRS_DEP_TIME', 'DEP_DELAY', 'TAXI_OUT', 'WHEELS_OFF', 'TAXI_IN', 'CRS_ARR_TIME']
target = 'ARR_DELAY'
X = Dataset[features].head(10000)
y = Dataset[target].head(10000)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
LO=LogisticRegression(max_iter=1000,random_state=42)
LO.fit(X_train,y_train)
```

```
LOy_pred=LO.predict(X_test)
print("Y_predict value=",LOy_pred)
LO_R2=r2_score(LOy_pred,y_test)*100
print("R2 square = ",LO_R2)
```

Y_predict value= [-2. -7. -11. ... -8. -15. -15.] R2 square = 79.14499343171778

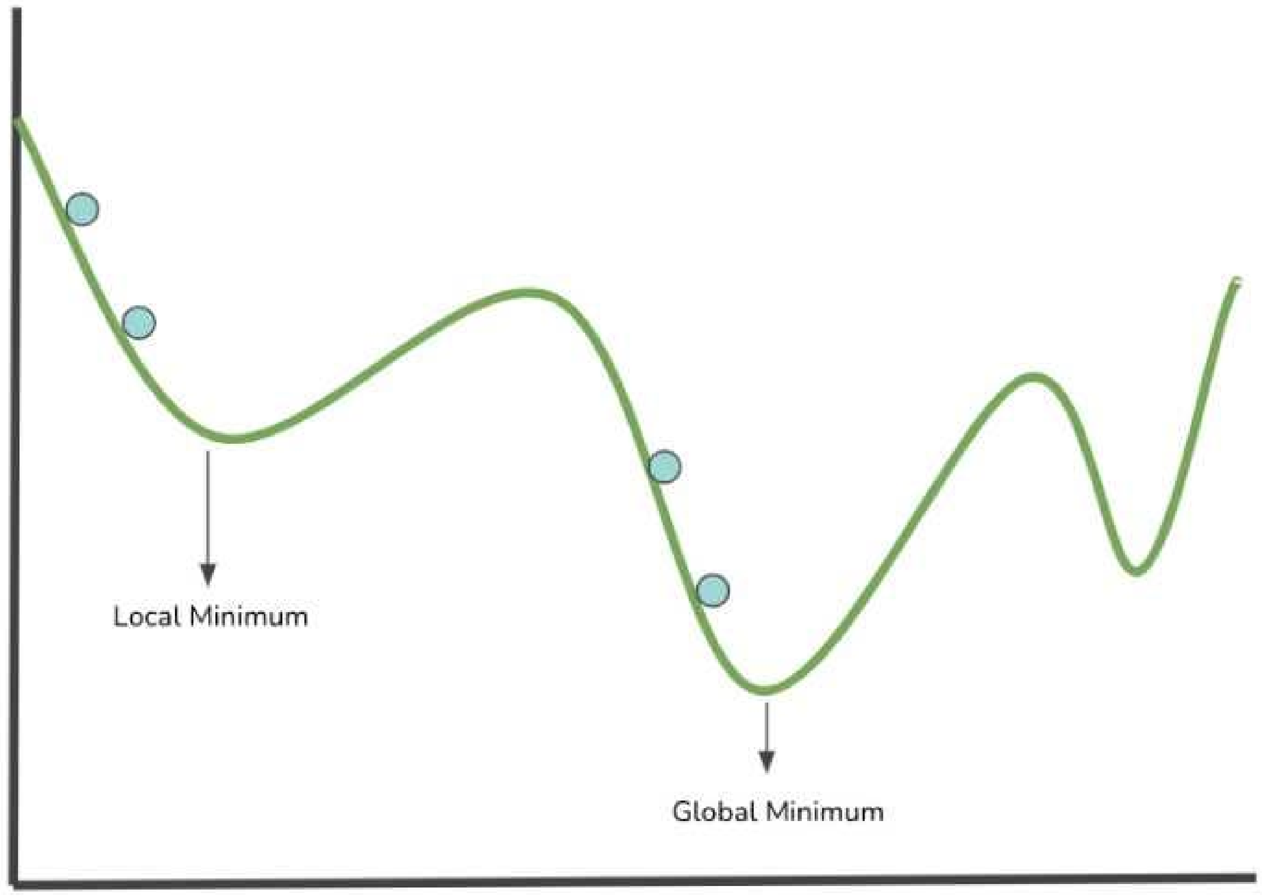
In [ ]:
```
if SGDR2>LR_R2 and SGDR2>LO_R2:
  print("We can decide that Stochastic Gradient Descent algorithm is best \nfor prediction an
else:
  print("We cannot say that Stochastic Gradient Descent algorithm is best \nfor prediction an
```

```
We can decide that Stochastic Gradient Descent algorithm is best
for prediction analysis compared to Linear Regression and Logistic Regression
```

PROJECT DESCRIPTION

Stochastic Gradient Descent

It is a machine learning optimization algorithm. Unlike traditional gradient descent, SGD updates a model's parameters using the gradient of the cost function for a randomly chosen individual data point (or a small batch). This stochastic approach makes it computationally efficient, particularly for large datasets, and allows for faster convergence. However, the randomness introduces noise, and careful tuning of the learning rate is required. SGD is widely used in training various machine learning models due to its efficiency and ability to handle large datasets.



Linear Regression

Linear Regression is a simple and widely-used statistical method in machine learning for predicting a continuous outcome variable based on one or more predictor variables. It assumes a linear relationship between the predictors and the target variable, represented by a straight line. The model learns the coefficients that minimize the difference between the predicted and actual values. Linear Regression is interpretable, easy to implement, and serves as a foundational technique for more complex models.

Logistic Regression

Logistic Regression is a statistical method used for binary classification problems, where the outcome variable is categorical with two classes (e.g., 0 or 1, true or false). Despite its name, it is a classification algorithm rather than a regression one. Logistic Regression models the probability of the binary outcome using a logistic function, which constrains the predicted values to be between 0 and 1. It is widely applied in various fields for tasks such as spam detection, medical diagnosis, and credit scoring due to its simplicity and effectiveness.

GOOGLE COLAB LINK-
https://colab.research.google.com/drive/19bnPNSbIq4mZN_A4pi6XBplO0iGQDTOc?usp=sharing

FUTURE DEVELOPEMENTS

```
• Conduct further analysis on the impact of external factors on flight
delays,allowing for more accurate predictions and insights.
• Incorporate additional features such as weather data and historical flight
information to enhance model accuracy.
• Investigate deep learning approaches, such as neural networks, for more
complex pattern recognition.
• Develop a user-friendly interface or application for easy accessibility
and utilization of the predictive models.
```

CONCLUSION

Thus successfully Implemented and compared Stochastic Gradient Descent (SGD), Linear Regression, and Logistic Regression models for flight arrival delay prediction, and found that Stochastic Gradient Descent (SGD) has more accuracy comapred to Linear Regression and Logistic Regression model.