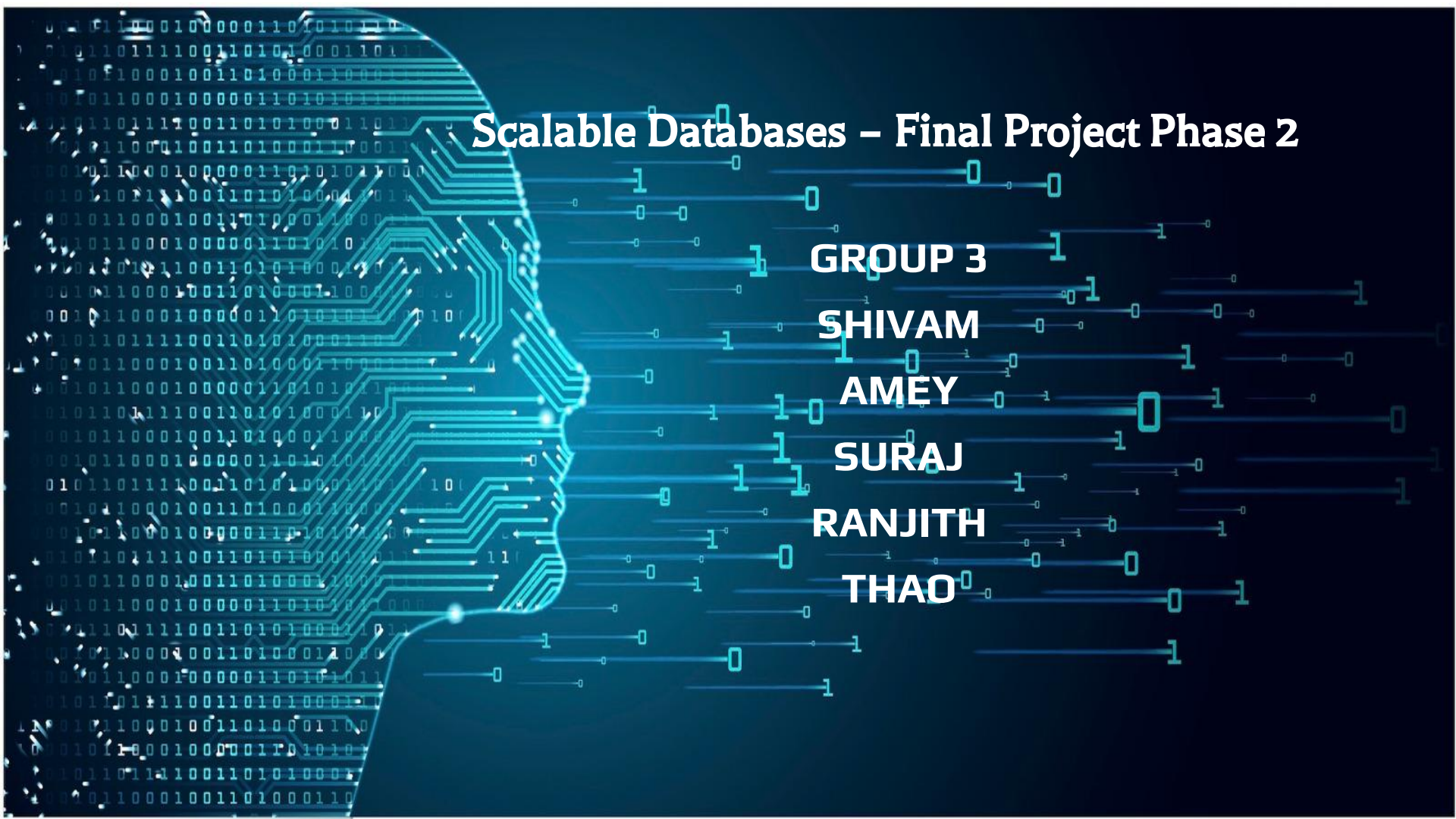
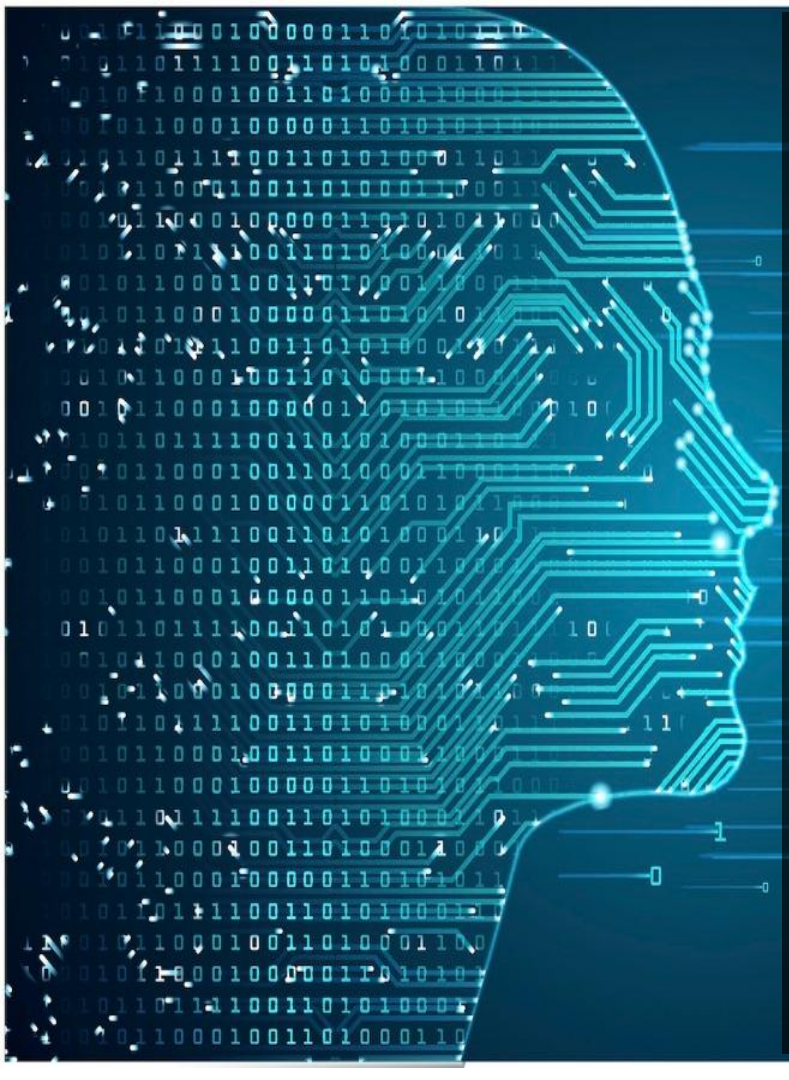
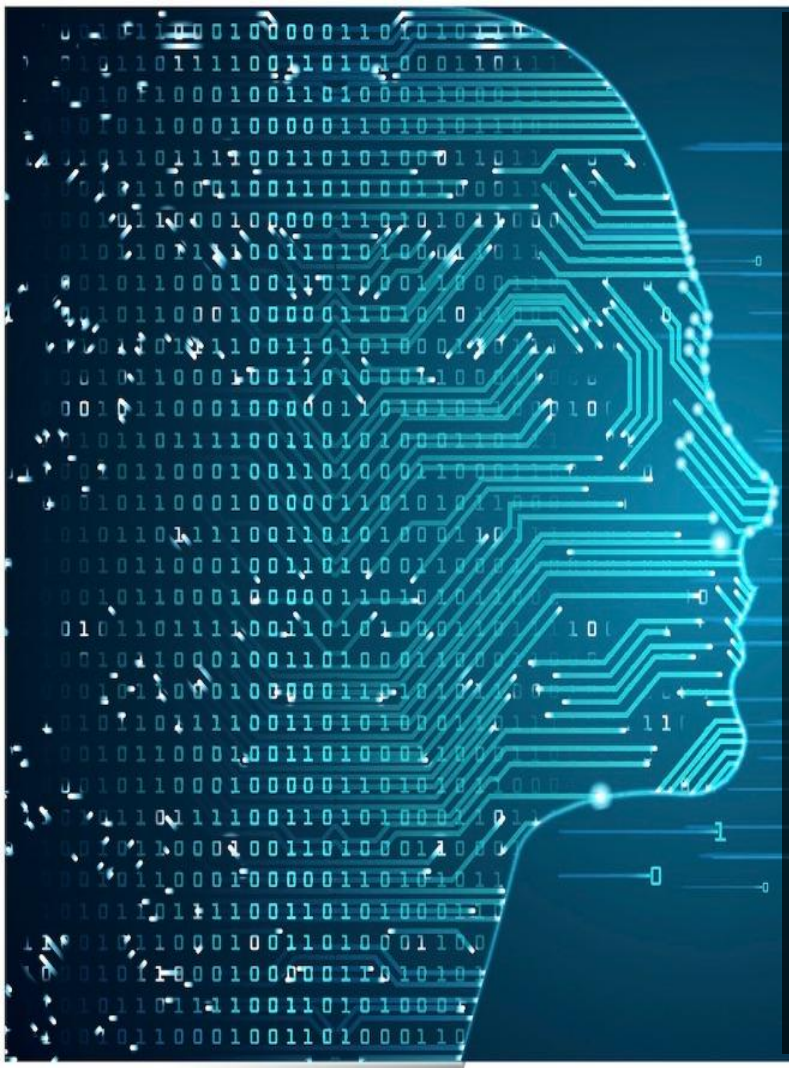

Scalable Databases – Final Project Phase 2
GROUP 3
SHIVAM
AMEY
SURAJ
RANJITH
THAO



# TABLES OF CONTENT:



# DATA SAMPLING IN HIVE

- Create new sample table
- Add “Delayed” column
- Extract 30,000 rows into a new table with random sampling
- Save to local machine as a .csv file

```
hive> CREATE TABLE IF NOT EXISTS AmeySamples (
    > Year INT,
    > Month INT,
    > DayofMonth INT,
    > DayOfWeek INT,
    > DepTime STRING,
    > CRSDepTime STRING,
    > ArrTime STRING,
    > CRSArrTime STRING,
    > UniqueCarrier STRING,
    > FlightNum INT,
    > TailNum INT,
    > ActualElapsedTime INT,
    > CRSElapsedTime INT,
    > AirTime INT,
    > ArrDelay INT,
    > DepDelay STRING,
    > Origin STRING,
    > Dest STRING,
    > Distance INT,
    > TaxiIn INT,
    > TaxiOut STRING,
    > Cancelled INT,
    > CancellationCode STRING,
    > Diverted INT,
    > CarrierDelay INT,
    > WeatherDelay INT,
    > NASDelay INT,
    > SecurityDelay INT,
    > LateAircraftDelay INT,
    > PRIMARY KEY (UniqueCarrier, Origin, Dest) DISABLE NOVALIDATE
    > )
    > COMMENT 'Flight Info'
    > ROW FORMAT SERDE 'org.apache.hadoop.hive.serde2.OpenCSVSerde'
    > WITH SERDEPROPERTIES (
    > "seperatorChar" = ","
    > );
OK
Time taken: 0.067 seconds
```

```
Time taken: 0.067 seconds
hive> INSERT INTO TABLE AmeySamples
    > SELECT * FROM Amey2002
    > DISTRIBUTE BY RAND()
    > SORT BY RAND()
    > LIMIT 30000;
Query ID = hadoop_20240503031111_9ca15f08-9692-405e-adfe-d3f72c90756f
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1714704766369_0001)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      4          4        0        0       0       0
Reducer 2 ...... container     SUCCEEDED      2          2        0        0       0       0
Reducer 3 ...... container     SUCCEEDED      1          1        0        0       0       0
Reducer 4 ...... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 04/04  [==========================>>] 100%  ELAPSED TIME: 45.04 s
--------------------------------------------------------------------------------
Loading data to table ameyflightinfo.ameysamples
OK
Time taken: 46.958 seconds
hive> Select count(1) from AmeySamples;
OK
30000
```

```
hive> CREATE TABLE temp AS
    > SELECT *,
    >        CASE
    >            WHEN ArrDelay <= 0 AND DepDelay <= 0 THEN 'N'
    >            ELSE 'Y'
    >        END AS Delayed
    > FROM AmeySamples;
Query ID = hadoop_20240503031415_485d3e0e-cb87-4cd6-a90c-3182d784436f
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1714704766369_0001)

--------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
--------------------------------------------------------------------------------
Map 1 .......... container     SUCCEEDED      1          1        0        0       0       0
--------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 4.74 s
--------------------------------------------------------------------------------
Moving data to directory hdfs://ip-172-31-13-69.ec2.internal:8020/user/hive/warehouse/ameyflightinfo.db/temp
OK
Time taken: 5.199 seconds
hive>
```

```
hive> exit;
[hadoop@ip-172-31-13-69 ~]$ hdfs dfs -mkdir selected_data
[hadoop@ip-172-31-13-69 ~]$ hive
Hive Session ID = b6382319-4406-46aa-adfa-38f7bc774d8a

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
hive> SET hive.cli.print.header=true;
hive> use AmeyFlightInfo;
OK
Time taken: 0.232 seconds
hive> INSERT OVERWRITE DIRECTORY 'hdfs:///user/hadoop/selected_data/'
    > ROW FORMAT DELIMITED
    > FIELDS TERMINATED BY ','
    > SELECT * FROM temp;
Query ID = hadoop_20240503032652_e3500c7b-1e69-4960-bd40-979366852e7d
Total jobs = 1
Launching Job 1 out of 1
Status: Running (Executing on YARN cluster with App id application_1714704766369_0002)

----------------------------------------------------------------------------------------------
        VERTICES      MODE        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED  KILLED
----------------------------------------------------------------------------------------------
Map 1 ..........   container     SUCCEEDED      1          1        0        0       0       0
----------------------------------------------------------------------------------------------
VERTICES: 01/01  [==========================>>] 100%  ELAPSED TIME: 4.83 s
----------------------------------------------------------------------------------------------
Moving data to directory hdfs:/user/hadoop/selected_data
OK
temp.year       temp.month      temp.dayofmonth temp.dayofweek  temp.deptime    temp.crsdeptime temp.arrtime    temp.crsarrtime temp.uniquecarrier  t
emp.flightnum   temp.tailnum    temp.actualelapsedtime  temp.crselapsedtime     temp.airtime    temp.arrdelay   temp.depdelay   temp.origin     temp.
dest    temp.distance   temp.taxiin     temp.taxiout    temp.cancelled  temp.cancellationcode   temp.diverted   temp.carrierdelay       temp.weatherd
elay    temp.nasdelay   temp.securitydelay      temp.lateaircraftdelay  temp.delayed
Time taken: 6.829 seconds
hive> exit;
[hadoop@ip-172-31-13-69 ~]$
```

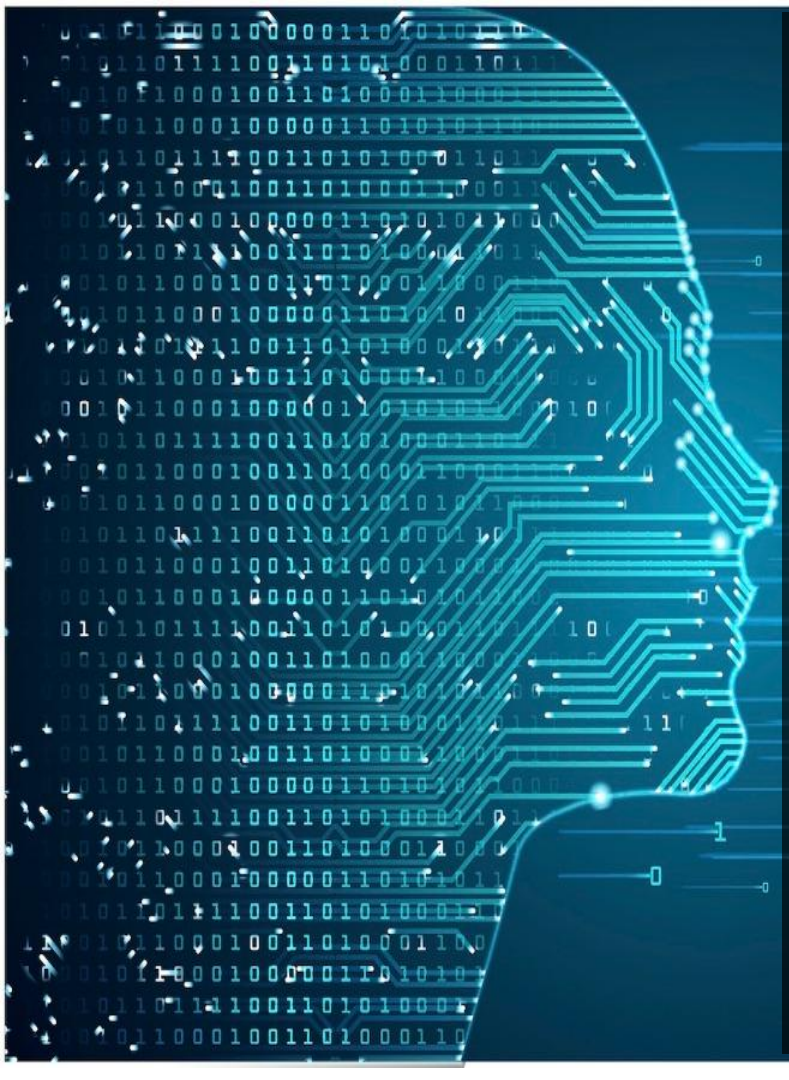
```
hive> exit;
[hadoop@ip-172-31-13-69 ~]$ hive -e 'SET hive.cli.print.header=true; use AmeyFlightInfo ; SELECT * FROM  temp LIMIT 0' \ | sed 's/[\]/,/g'> /home/had
oop/header.csv
Hive Session ID = d591dab3-d3fc-4106-a08e-a4ad384b39ca

Logging initialized using configuration in file:/etc/hive/conf.dist/hive-log4j2.properties Async: false
OK
Time taken: 0.741 seconds
OK
Time taken: 1.584 seconds
[hadoop@ip-172-31-13-69 ~]$ hadoop fs -get /user/hadoop/selected_data/* /home/hadoop/data
[hadoop@ip-172-31-13-69 ~]$ cat /home/hadoop/header.csv /home/hadoop/data > /home/hadoop/2002_data.csv
[hadoop@ip-172-31-13-69 ~]$
```

MINGW64:/c/Users/Amey Borkar

```
Amey Borkar@AmeyBorkar MINGW64 ~ (master)
$ chmod 400 "C:\Users\Amey Borkar\Downloads\Scalable_Phase_2.pem"
```

Administrator: Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\WINDOWS\system32> scp -i "C:\\Users\\Amey Borkar\\Downloads\\Scalable_Phase_2.pem" hadoop@ec2-3-231-222-203.comput
e-1.amazonaws.com:/home/hadoop/2002_data.csv  "C:\\Users\\Amey Borkar\\Downloads\\"
2002_data.csv                                                 100% 3013KB    6.1MB/s   00:00
```



# DATA PREPARATION IN PYTHON

1. Combining sample data from all years
2. Finding the null value columns from the merged data
3. Dropping column with more null value
4. Transformations:
   4.1 Numerical columns are replaced with Median of the null value columns
   4.2 Categorical data are replaced with the Mode data of the column
   4.3 Encode Categorical values with LabelEncoder
5. Split the data (Train, Validate, Test)

```python
# Read each CSV file into separate DataFrames
df1 = pd.read_csv('/content/1996_data.csv')
df2 = pd.read_csv('/content/2000_data.csv')
df3 = pd.read_csv('/content/2002_data.csv')
df4 = pd.read_csv('/content/2003_data.csv')
df5 = pd.read_csv('/content/2007_data.csv')

# Merge the DataFrames
merged_df = pd.concat([df1, df2, df3, df4, df5], ignore_index=True)

# Save the merged DataFrame to a new CSV file
merged_df.to_csv('merged_file.csv', index=False)
```

```python
#Droping columns with a large number of missing values
df.drop(columns=['carrierdelay', 'weatherdelay',
                 'nasdelay', 'securitydelay',
                 'lateaircraftdelay','cancellationcode',
                 'arrdelay', 'depdelay'], inplace=True)
#To Check if there are any remaining missing values
print(df.isnull().sum())
```

```python
# Calculate the number of null values in each column
null_values = merged_df.isnull().sum()
null_values
```

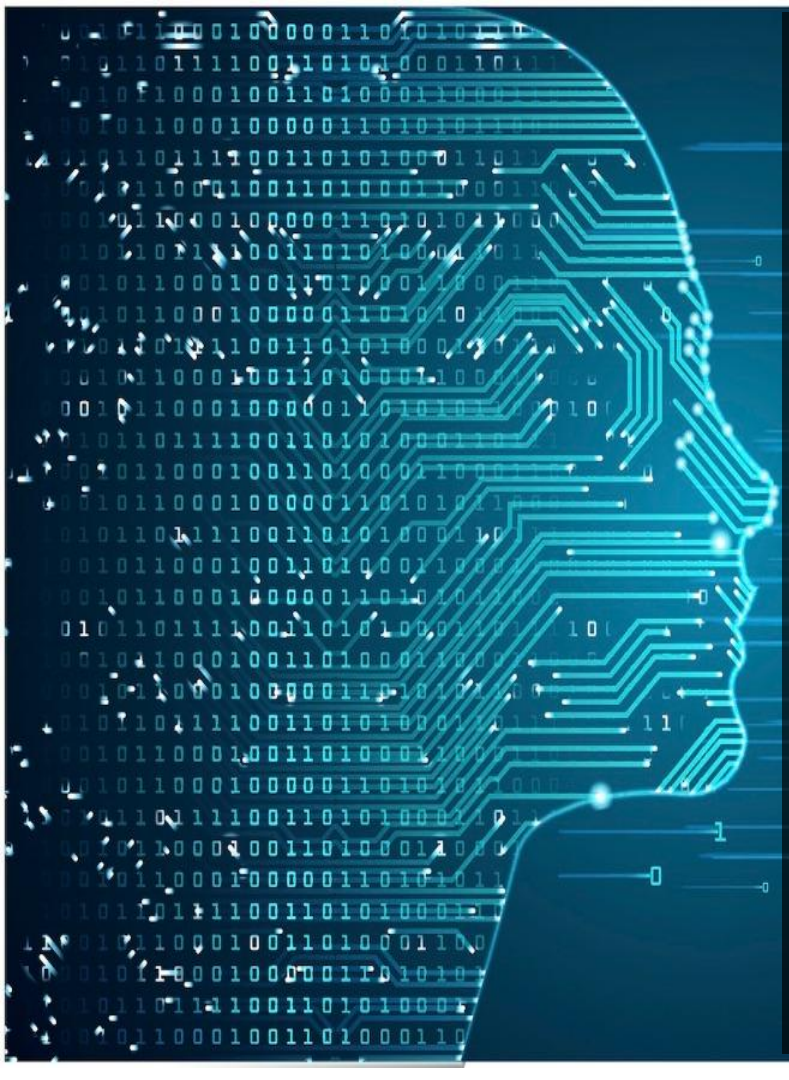
```
year                    0
month                   0
dayofmonth              0
dayofweek               0
deptime              3167
crsdeptime              0
arrtime              3495
crsarrtime              0
uniquecarrier           0
flightnum               0
tailnum               284
actualelapsedtime    3495
crselapsedtime         23
airtime              3495
arrdelay             3495
depdelay             3167
origin                  0
dest                    0
distance                0
taxiin                  0
taxiout                 0
cancelled               0
cancellationcode   149124
diverted                0
carrierdelay       102288
weatherdelay       102288
nasdelay           102288
securitydelay      102288
lateaircraftdelay  102288
delayed                 0
dtype: int64
```

```python
from sklearn.impute import SimpleImputer

# Separate numerical and categorical columns
numerical_cols = merged_df.select_dtypes(include=['float64', 'int64']).columns
categorical_cols = merged_df.select_dtypes(include=['object']).columns

# Impute missing values for numerical columns with the median
numerical_imputer = SimpleImputer(strategy='median')
merged_df[numerical_cols] = numerical_imputer.fit_transform(merged_df[numerical_cols])

# Impute missing values for categorical columns with the most frequent category
categorical_imputer = SimpleImputer(strategy='most_frequent')
merged_df[categorical_cols] = categorical_imputer.fit_transform(merged_df[categorical_cols])

# Check if there are any remaining missing values
print(merged_df.isnull().sum())

# Now all missing values have been imputed.
```

```python
df_encoded = df.copy()

label_encoder = LabelEncoder()
for column in df_encoded.columns:
    if df_encoded[column].dtype == 'object':  #To Check if the column is categorical
        df_encoded[column] = label_encoder.fit_transform(df_encoded[column])
```

```python
#Spliting the dataset into Training and Testing sets
X = df_encoded.drop(columns=['delayed'])
y = df_encoded['delayed']
```

```python
# split the data in train validate test in 70:15:15

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.15, random_state=123)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.15, random_state=123)
```



# MODEL IMPLEMENTATION

- XGBoost
- Gradient Boosting
- Decision Trees
- Random Forest
- Logistic Regression

# XGBOOST

```python
import xgboost as xgb

# Initialize and train the XGBoost model
model = xgb.XGBClassifier(
    learning_rate=0.01,
    max_depth=15,
    n_estimators=1500,
    subsample=0.8,
    colsample_bytree=0.8,
    reg_alpha=0.1,
    reg_lambda=0.1,
    gamma=0.1,
    objective='binary:logistic'
)

model.fit(X_train, y_train)

# Make predictions on the test set
predictions = model.predict(X_test)
```

```
Accuracy Score: 0.9031111111111111
Precision Score: 0.9280486802072198
Confusion Matrix:
[[ 9034   875]
 [ 1305 11286]]
Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.91      0.89      9909
           1       0.93      0.90      0.91     12591

    accuracy                           0.90     22500
   macro avg       0.90      0.90      0.90     22500
weighted avg       0.90      0.90      0.90     22500
```

# Gradient Boosting:

```
gradient_boost = GradientBoostingClassifier(
    n_estimators=100,
    learning_rate=0.1,
    max_depth=3,
    min_samples_split=2,
    min_samples_leaf=1,
    random_state=42
)

# Train the classifier
gradient_boost.fit(X_train, y_train)

# Make predictions on the validation set
predictions_val = gradient_boost.predict(X_val)
```

```
Validation Accuracy: 0.7277908496732026
Test Accuracy: 0.7233777777777778
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.66      0.68      9909
           1       0.74      0.77      0.76     12591

    accuracy                           0.72     22500
   macro avg       0.72      0.72      0.72     22500
weighted avg       0.72      0.72      0.72     22500

Confusion Matrix:
[[6524 3385]
 [2839 9752]]
```

# Decision Tree:

```python
# Initialize the decision tree classifier
decision_tree = DecisionTreeClassifier(random_state=123)

# Train the decision tree classifier on the training set
decision_tree.fit(X_train, y_train)

# Make predictions on the validation set
y_pred_val = decision_tree.predict(X_val)

# Calculate the accuracy score for the validation set
accuracy_val = accuracy_score(y_val, y_pred_val)
print("Validation Accuracy:", accuracy_val)

# Fine-tune the decision tree classifier using hyperparameter
param_grid = {
    'max_depth': [None, 5, 10],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

grid_search = GridSearchCV(decision_tree, param_grid, cv=3)
grid_search.fit(X_train, y_train)

# Make predictions on the validation set using the best model
y_pred_val = grid_search.best_estimator_.predict(X_val)
```

```
Validation Accuracy: 0.7178562091503268
Validation Accuracy after hyperparameter tuning: 0.7328627450980392
Validation Precision: 0.7755635097223565
Test Accuracy after hyperparameter tuning: 0.734
Test Precision: 0.774975024975025
Confusion Matrix for Test Set:
[[7206 2703]
 [3282 9309]]
Classification Report for Test Set:
              precision    recall  f1-score   support

           0       0.69      0.73      0.71      9909
           1       0.77      0.74      0.76     12591

    accuracy                           0.73     22500
   macro avg       0.73      0.73      0.73     22500
weighted avg       0.74      0.73      0.73     22500
```

# Random Forest

```python
random_forest = RandomForestClassifier(
    n_estimators=200,
    max_depth=10,
    min_samples_split=4,
    min_samples_leaf=2,
    random_state=84
)

# Train the classifier
random_forest.fit(X_train, y_train)

# Make predictions on the validation set
predictions_val = random_forest.predict(X_val)
```

```
Validation Accuracy: 0.6993986928104575
Test Accuracy: 0.6986666666666667
Classification Report:
              precision    recall  f1-score   support

           0       0.69      0.57      0.63      9909
           1       0.70      0.80      0.75     12591

    accuracy                           0.70     22500
   macro avg       0.70      0.69      0.69     22500
weighted avg       0.70      0.70      0.69     22500

Confusion Matrix:
[[ 5697  4212]
 [ 2568 10023]]
```

# Logistic Regression

```
logistic_regression = LogisticRegression(
    C=2.0,
    penalty='l2',
    solver='newton-cg',
    max_iter=200,
    random_state=123
)

# Train the model on the training set
logistic_regression.fit(X_train, y_train)

# Make predictions on the validation set
y_pred_val = logistic_regression.predict(X_val)
```

```
Validation Accuracy: 0.7178562091503268
Validation Precision: 0.7465366387167335
Test Accuracy: 0.7186222222222223
Test Precision: 0.7472353870458136
Confusion Matrix for Test Set:
[[6709 3200]
 [3131 9460]]
Classification Report for Test Set:
              precision    recall  f1-score   support

           0       0.68      0.68      0.68      9909
           1       0.75      0.75      0.75     12591

    accuracy                           0.72     22500
   macro avg       0.71      0.71      0.71     22500
weighted avg       0.72      0.72      0.72     22500
```

# TARGET PREDICTIONS

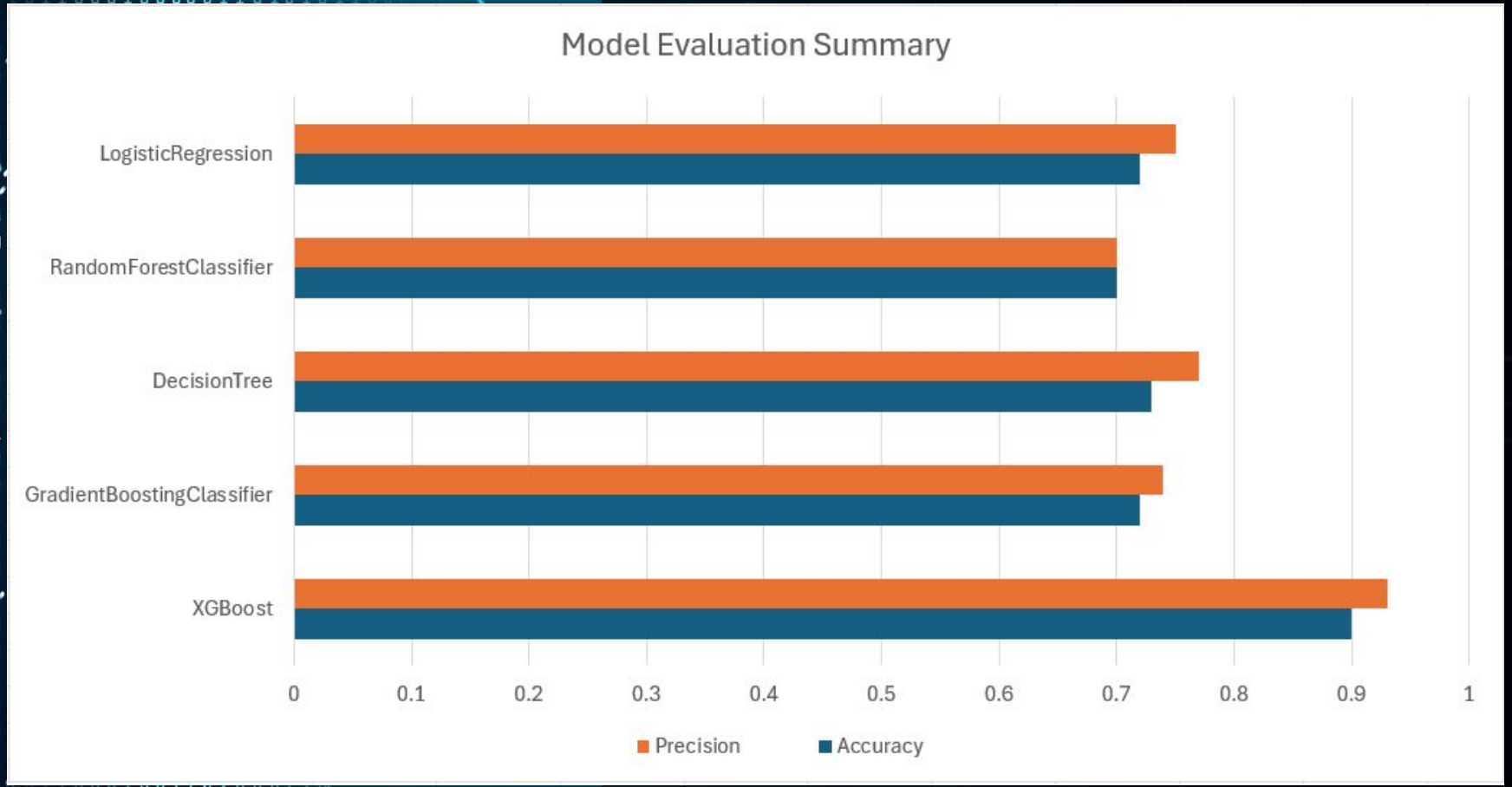
```
predictions_df_1['predicted_delayed'] = predictions_df_1['predicted_delayed'].map({1: 'Y', 0: 'N'})
```

```
output_df_1 = pd.DataFrame(predictions_df_1['predicted_delayed'])
output_df_2 = pd.DataFrame(predictions_df_2['predicted_delayed'])
output_df_3 = pd.DataFrame(predictions_df_3['predicted_delayed'])
output_df_4 = pd.DataFrame(predictions_df_4['predicted_delayed'])
output_df_5 = pd.DataFrame(predictions_df_5['predicted_delayed'])

output_df = pd.concat([output_df_1, output_df_2, output_df_3, output_df_4, output_df_5], axis=1)

output_df.columns = ['XGBoost','GradientBoosting','DecisionTree','RandomForest','LogisticRegression']

output_df
```

# TARGET PREDICTIONS

| | XGBoost | GradientBoosting | DecisionTree | RandomForest | LogisticRegression |
|---|---|---|---|---|---|
| 0 | Y | Y | Y | Y | Y |
| 1 | N | N | N | Y | N |
| 2 | N | N | Y | N | N |
| 3 | N | Y | Y | Y | N |
| 4 | N | N | N | Y | N |
| 5 | Y | Y | N | Y | N |
| 6 | N | N | Y | N | N |
| 7 | Y | Y | Y | Y | Y |
| 8 | N | N | N | Y | N |
| 9 | Y | Y | Y | Y | Y |

# MODEL EVALUATIONS



## CONCLUSION:

- XGBoost performs the best out of the 5 models
- ~90% accuracy on test dataset
- Model is predicting accurate values based on given features.

# Extra step to check model results

- Create depdelay and arrdelay column using arrtime vs. crsarrtime and deptime vs. crsdeptime

| | arrdelay | depdelay | predicted_delayed |
|---|---|---|---|
| 0 | 20.0 | -1.0 | Y |
| 1 | -8.0 | 0.0 | N |
| 2 | -76.0 | -12.0 | N |
| 3 | -8.0 | 0.0 | Y |
| 4 | -5.0 | 0.0 | N |
| 5 | 4.0 | 5.0 | Y |
| 6 | -10.0 | -5.0 | N |
| 7 | 1.0 | 3.0 | Y |
| 8 | -5.0 | 0.0 | N |
| 9 | 1.0 | -1.0 | Y |

```
predictions_df = predictions_df_1.copy()
predictions_df['depdelay'] = predictions_df['deptime'] - predictions_df['crsdeptime']
predictions_df['arrdelay'] = predictions_df['arrtime'] - predictions_df['crsarrtime']
predictions_df
```

Extra Credits?


THANK YOU