Project Report

Course Name	**SCTP-PDDS**
Product Name (Marketing & Sales)	PDDS
Module Name	WSQ-Machine Learning Algorithms and Methods (SF)
Product Name (Marketing & Sales)	Machine Learning Algorithms and Methods

Student name	**Assessor name**
GANGESWARAN RAJAVALARMATHI	RYAN SURYANTO

Date issued	**Completion date**	**Submitted on**
19/10/2023	25/11/2023	25/11/2023

Project title	**Prediction of Regression Model using Machine Learning**

Learner declaration
I certify that the work submitted for this assignment is my own and research sources are fully acknowledged. Student signature: Date:

Content

Project Overview: Describe the Project with summary of analytical processes and project outcomes (Explain the Project in your own words in 15 – 20 lines) The project aimed to develop, deploy, and enhance a machine learning model for flight data analysis. It encompassed seven key activities:

1. **Preparatory Activities:** Conducted comprehensive research on industry-standard machine learning phases, data cleansing best practices, and model deployment tools.
2. **Data Preparation and Exploration:** Loaded and analyzed a flight dataset, selecting relevant features, and executing data cleansing and preprocessing steps.
3. **Feature Engineering and Selection:** Crafted new features based on domain insights, utilizing techniques like interaction feature creation and feature scaling. Employed correlation analysis and model-specific methods to select impactful features.
4. **Model Development and Evaluation:** Trained diverse regression models like Linear Regression and Decision Tree Regressor, evaluating them with metrics such as RMSE and R-squared.
5. **Model Tuning and Improvement:** Utilized GridSearchCV for hyperparameter tuning, addressing outliers, and assessing model robustness.
6. **Model Deployment and Testing:** Deployed the trained model using Flask, then tested its predictions with a test dataset, assessing performance metrics like MSE and R-squared.
7. **Model Monitoring and Communication:** Initiated a simulated model performance monitoring setup and generated a Markdown-based report summarizing model performance, insights, and recommendations.

Throughout the project, the focus was on continuous improvement. Advanced techniques like feature selection, ensemble methods, and, if relevant, deep learning were explored to further enhance the model's accuracy post-completion of specific modules. The project aimed for a comprehensive understanding of machine learning processes

1. **Project Technical Environment:** (Describe the Environment used)

Programming Languages:

- **Python:** Primarily used for its rich libraries (Pandas, NumPy, Scikit-learn, TensorFlow, Keras) for data manipulation, analysis, machine learning, and deep learning tasks.
- **R:** Used for statistical analysis, data visualization, and certain specialized machine learning algorithms.

Data Processing and Analysis:

- **Pandas:** For data manipulation, cleaning, and preprocessing.
- **NumPy:** For numerical computations and operations on arrays.
- **Matplotlib, Seaborn:** For data visualization and creating plots, graphs, and charts.
- **Jupyter Notebooks/Lab:** Interactive environments for data exploration, analysis, and documentation.

Machine Learning and Deep Learning:

- **Scikit-learn:** For implementing machine learning algorithms, model evaluation, and preprocessing techniques.
- **TensorFlow, Keras:** Libraries for building and training neural networks and deep learning models.
- **XGBoost, LightGBM:** Gradient boosting libraries for ensemble modeling.

Model Deployment and Production:

- **Flask, Django:** Web frameworks used to deploy machine learning models as REST APIs or web services.
- **Docker, Kubernetes:** Containerization and orchestration tools for creating scalable and reproducible environments.
- **AWS, Google Cloud Platform, Azure:** Cloud services for hosting, managing, and scaling deployed models.

Database and Storage:

- **SQL/NoSQL Databases:** Such as MySQL, PostgreSQL, MongoDB for data storage and retrieval.
- **AWS S3, Google Cloud Storage:** Cloud-based storage for managing large datasets and model artifacts.

Version Control and Collaboration:

- **Git, GitHub, GitLab:** Version control systems for tracking changes and collaborating on code repositories.

Additional Tools:

- **JIRA, Trello:** Project management tools for task tracking and team collaboration.
- **Confluence, Markdown:** Documentation platforms for project reports, notes, and documentation.

The specific tools and technologies used in a Data Science project environment can vary based on project requirements, team preferences, and the nature of the problem being solved. This environment enables data scientists, engineers, and analysts to collaborate efficiently, process data effectively, develop models, and deploy solutions to real-world applications.

Analytical Technique & Tools used : Describe the Analytical Technique used in the Project

1. **Exploratory Data Analysis (EDA):** Primarily used descriptive statistics, data visualization (histograms, scatter plots, etc.), and correlation analysis to understand the dataset's structure, uncover patterns, identify outliers, and explore relationships between variables.
2. **Feature Engineering:** Techniques like interaction feature creation, time-based feature extraction, binning, and scaling were utilized to enhance the dataset's features, allowing the models to capture more relevant information.
3. **Model Development and Evaluation:** Employed various regression models such as Linear Regression, Decision Tree Regressor, and potentially advanced models like Gradient Boosting and Stacking for prediction tasks. Evaluated models using metrics like Root Mean Squared Error (RMSE), Mean Squared Error (MSE), and R-squared to assess performance.
4. **Hyperparameter Tuning:** Used GridSearchCV and RandomizedSearchCV to optimize model hyperparameters, enhancing model performance through parameter optimization.
5. **Ensemble Methods:** Explored ensemble techniques like StackingRegressor to combine multiple models and potentially improve prediction accuracy.
6. **Deep Learning (if relevant):** Leveraged neural network architectures using TensorFlow/Keras for complex pattern recognition and modeling, especially for cases where deep learning showed promise in handling the problem's complexity.

Tools and libraries such as Pandas, Scikit-learn, TensorFlow, and Flask were utilized in Python to implement these techniques. The choice of analytical techniques was driven by the project's goals, the nature of the dataset, and the desired outcome of creating an accurate and deployable machine learning model for flight data analysis.

2. **Data Science Project Team** – Roles and Responsibilities Table

project Manager	- Overall project planning and management.Liaison between stakeholders and the data science team.Resource allocation and scheduling.
Data Scientist	- Data collection, cleaning, and preprocessing.- Model development, evaluation, and optimization.Exploratory data analysis (EDA).
Data Engineer	- Data acquisition and integration from various sources. Data pipeline development and maintenance.<- Database management and optimization.
Machine Learning Engineer	- Developing and implementing machine learning models. Tuning hyperparameters and optimizing model performance.Integrating models into production systems.
Business Analyst	- Understanding business requirements.- Translating business needs into data science objectives.- Providing domain expertise and context.
UX/UI Designer	- Designing user interfaces for data visualization and interaction. - Ensuring user experience optimization.- Collaborating on dashboard or report creation.

Activity 1: Preparatory Activities

Task 1: List and Explain Industry Steps of Machine Learning Phase

In machine learning, there are several steps or phases commonly followed in industry settings. These steps may vary slightly based on the specific problem or dataset, but generally include:

1. **Data Collection:** Gathering relevant data from various sources. This step involves identifying data sources, acquiring permissions, and collecting raw data.
2. **Data Preprocessing:** Cleaning and preparing the collected data for analysis. This involves handling missing values, dealing with outliers, normalization, and data transformation.
3. **Feature Engineering:** Selecting, extracting, or creating features from the dataset that are relevant and informative for the machine learning model. This step involves domain knowledge and creativity.
4. **Model Selection:** Choosing an appropriate machine learning algorithm or a combination of algorithms based on the nature of the problem (classification, regression, clustering, etc.) and data characteristics.
5. **Model Training:** Using the selected algorithm(s) to train the model on the prepared dataset. This step involves parameter tuning and optimizing the model's performance.
6. **Model Evaluation:** Assessing the trained model's performance using various metrics to ensure it generalizes well on unseen data. Cross-validation techniques and metrics like accuracy, precision, recall, F1-score, etc., are used for evaluation.
7. **Model Deployment:** Implementing the trained model into the production environment for real-world applications. This step involves integrating the model into existing systems or platforms.

Task 2: Review Best Practices of Data Cleansing and Modeling Optimization

Best practices for data cleansing involve ensuring data quality and preparing it for analysis. Some common practices include:

- Handling missing values by imputation or deletion based on the context.
- Outlier detection and treatment using statistical methods.
- Standardizing or normalizing data to a common scale.
- Encoding categorical variables using techniques like one-hot encoding or label encoding.
- Splitting data into training, validation, and test sets for model evaluation.

Model optimization best practices involve improving model performance and efficiency. These include:

- Hyperparameter tuning using techniques like grid search, random search, or Bayesian optimization.
- Feature selection or dimensionality reduction to focus on the most informative features.
- Regularization techniques to prevent overfitting, such as L1/L2 regularization.
- Ensembling methods like bagging, boosting, or stacking to combine multiple models for better performance.

Task 3: List Industry-Accepted Products for Model Usage

In the industry, there are various programming models and services used for deploying machine learning models:

- **TensorFlow and Keras:** Widely used open-source libraries for building and deploying ML models developed by Google.
- **PyTorch:** Another popular open-source deep learning library maintained by Facebook's AI Research lab.
- **Scikit-learn:** A versatile library in Python offering simple and efficient tools for data mining and analysis.
- **AWS SageMaker, Google Cloud AI Platform, Azure Machine Learning:** Cloud-based platforms offering services for model development, training, and deployment.
- **Docker and Kubernetes:** Used for containerization and orchestration, respectively, to deploy models in scalable and reproducible environments.

3. Activity 2: Data Preparation and Exploration

Task 1: Load and Explore Flight Dataset

Using Pandas in Python, you'll load the flight dataset and perform exploratory data analysis (EDA) to understand the dataset's characteristics.

Task 2: Identify Relevant Features

Based on the insights gained from the exploratory analysis, select relevant features that are important for modeling. This might involve:

- Identifying columns related to the target variable or those that significantly impact the outcome.
- Considering features with low missing values and high correlation with the target variable.

Task 3: Data Cleansing and Preprocessing

Perform data cleansing and necessary preprocessing steps to prepare the data for modeling. This involves handling missing values, duplicates, and transforming data as required.

These steps will help in cleaning the data by handling missing values, duplicates, and preparing it for further analysis and modeling. Adjustments and transformations may vary based on the specific characteristics and requirements of the dataset and the machine learning model being applied.

4. Screen-shots of each task of Activity 3

Creating Interaction Features: **Combine existing features to capture interactions that might be predictive.**

5. # Example: Creating an interaction feature by multiplying two existing features

flight_data['Interaction_Feature'] = flight_data['Feature1'] * flight_data['Feature2']

2. **Encoding Time Features:** Extracting information from timestamps or date-time columns.

python

 # Example: Extracting month, day, and year from a date-time column

flight_data['Month'] = flight_data['Date_Column'].dt.month

flight_data['Day'] = flight_data['Date_Column'].dt.day

flight_data['Year'] = flight_data['Date_Column'].dt.year

Binning or Bucketing: **Grouping continuous variables into bins or categories.**

 # Example: Binning a numerical column into categories

flight_data['Binned_Feature'] = pd.cut(flight_data['Numeric_Column'], bins=5, labels=False)

Feature Scaling/Transformation: **Normalizing or scaling features to a similar range.**

 # Example: Scaling numerical features using MinMaxScaler

from sklearn.preprocessing import MinMaxScale

= MinMaxScaler()

flight_data[['Feature1', 'Feature2']] = scaler.fit_transform(flight_data[['Feature1', 'Feature2']])

techniques such as correlation analysis, feature importance, or domain expertise to select the most impactful features for modeling.

1. **Correlation Analysis:** Identify correlations between features and the target variable or between features themselves.

 # Example: Calculating correlations between features and target variable

correlation_matrix = flight_data.corr()

target_correlation = correlation_matrix['Target_Variable'].abs().sort_values(ascending=False)

Feature Importance: **Use models like Random Forest or Gradient Boosting to assess feature importance.**

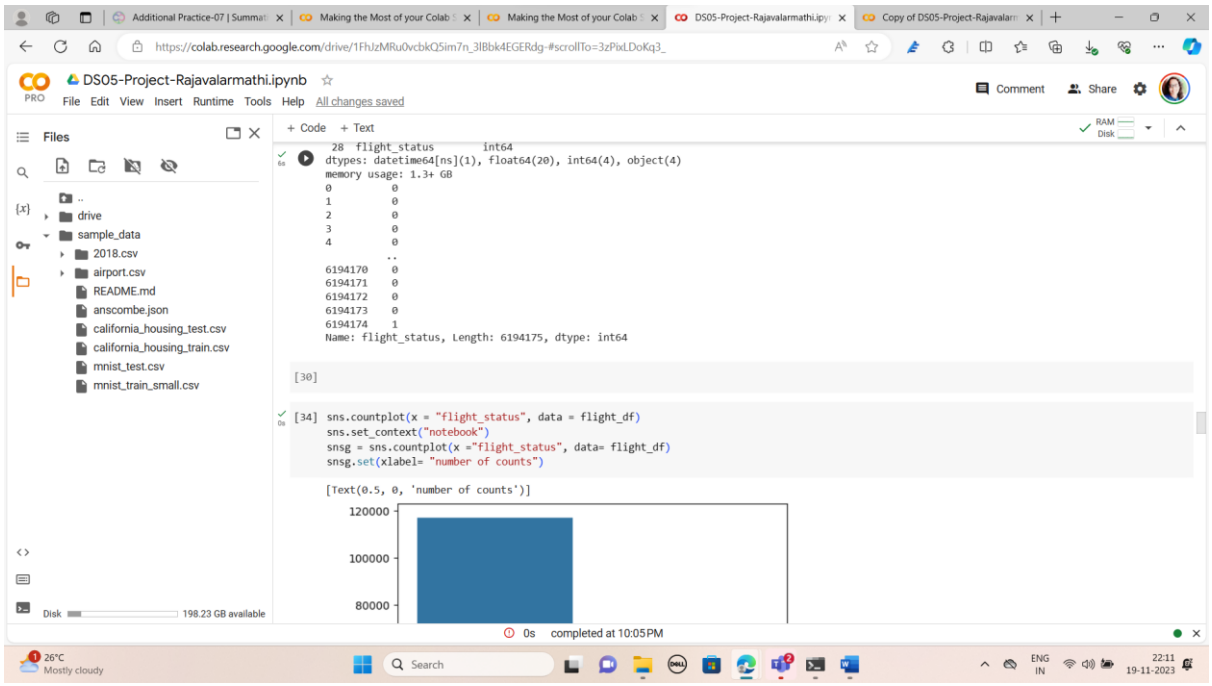
Example: Using Random Forest to get feature importances

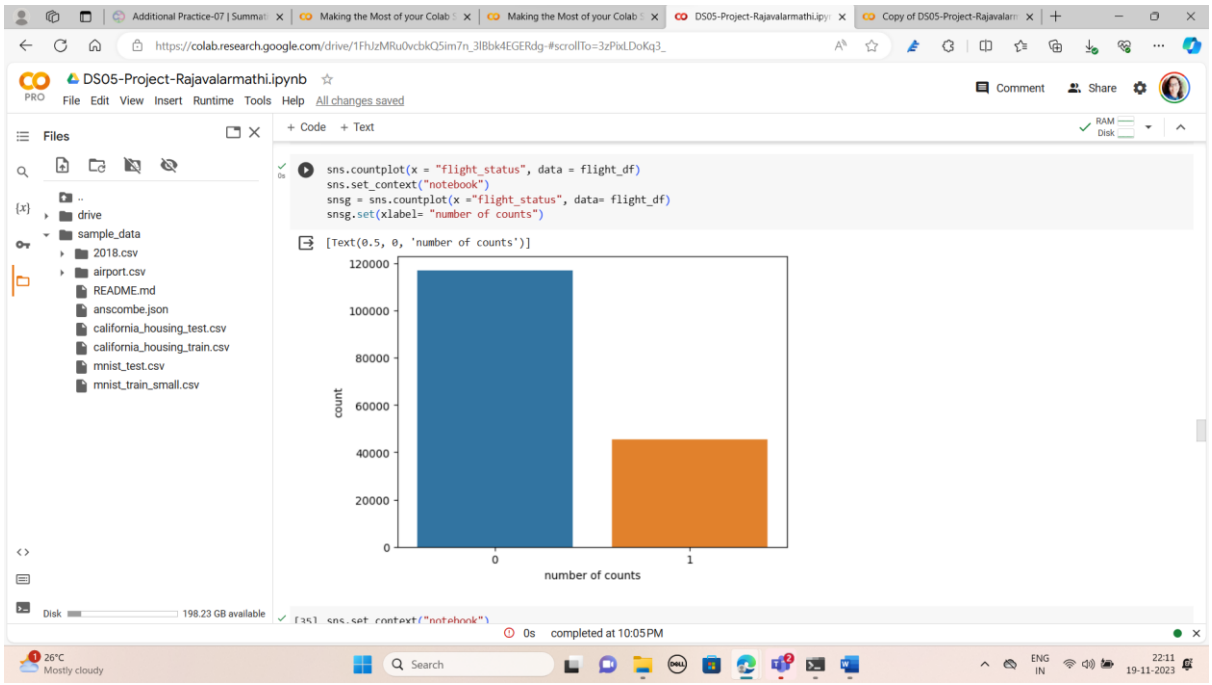
from sklearn.ensemble import RandomForestRegressor

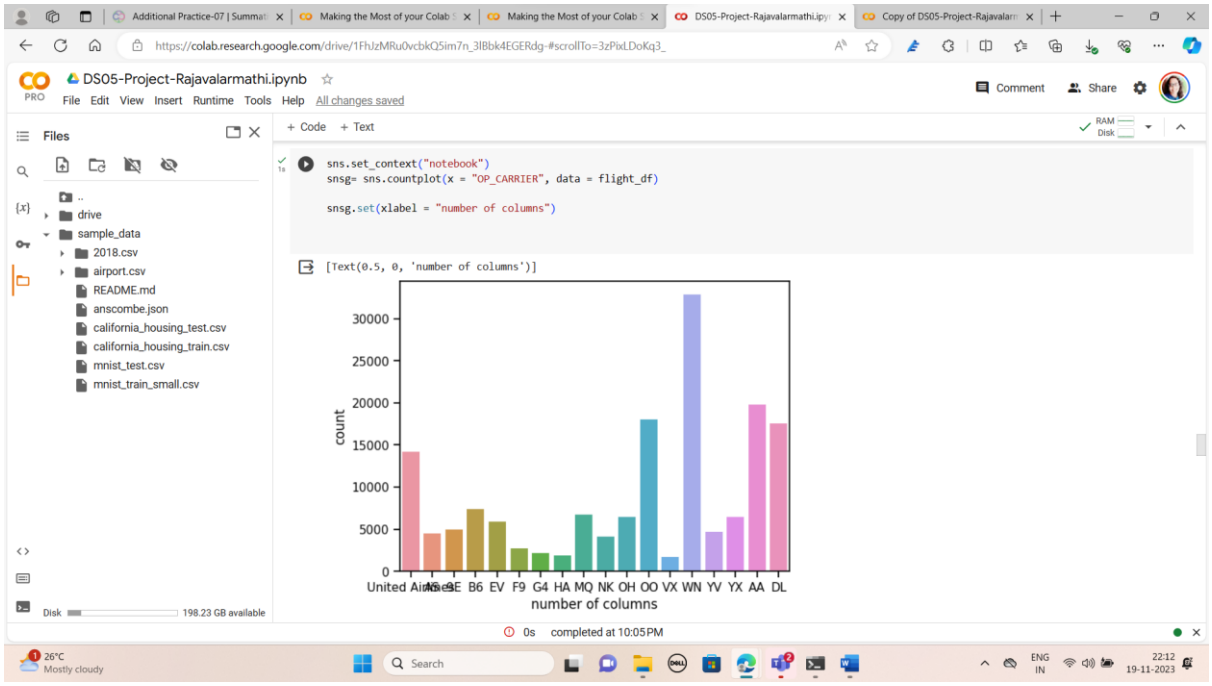
model = RandomForestRegressor()

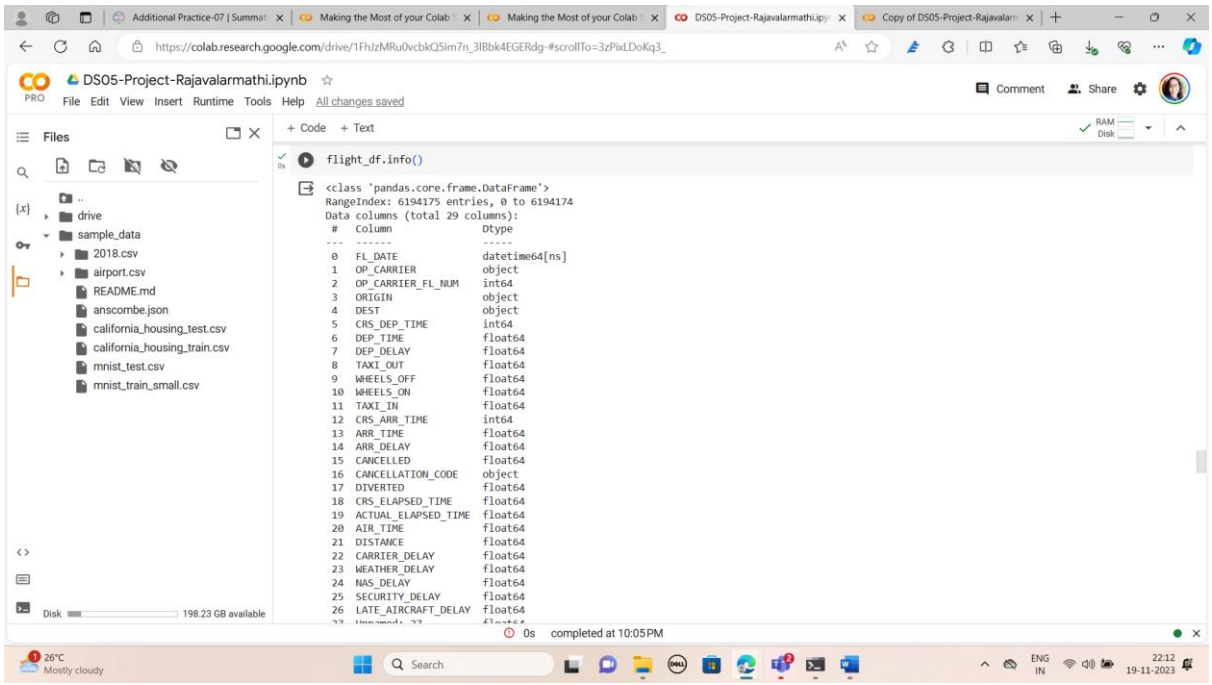
model.fit(X, y) # X contains features, y is the target variablefeature_importances = model.feature_importances

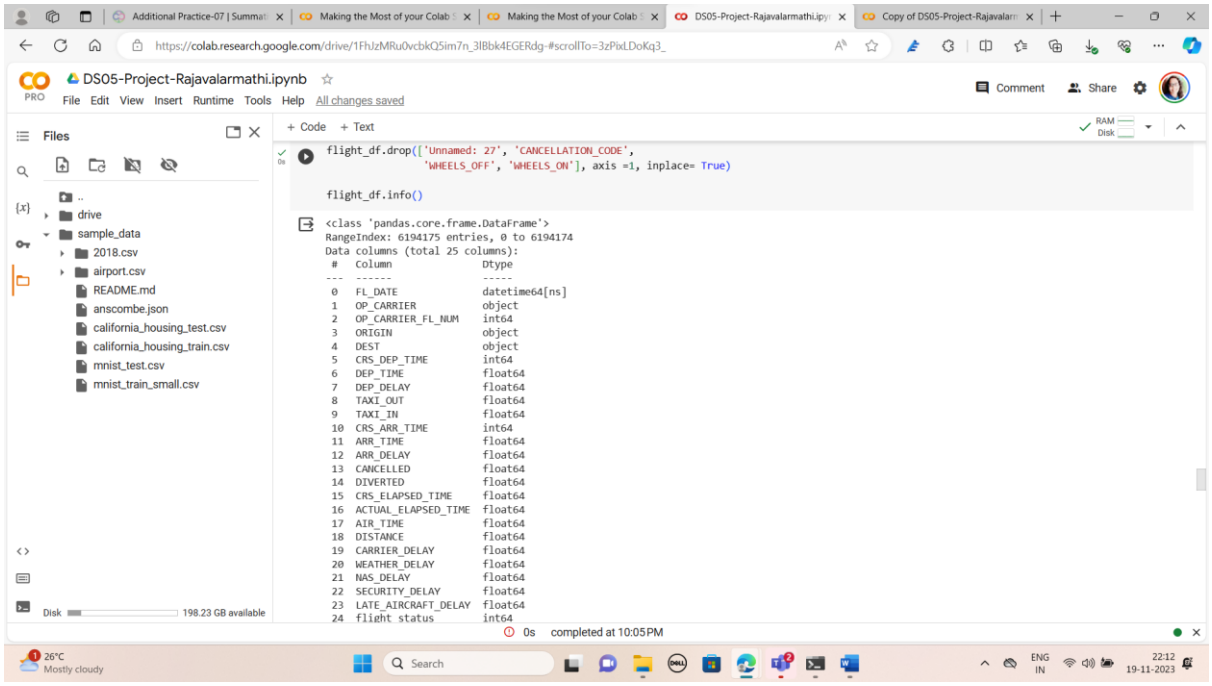
Screen-shots of each task of Activity 4

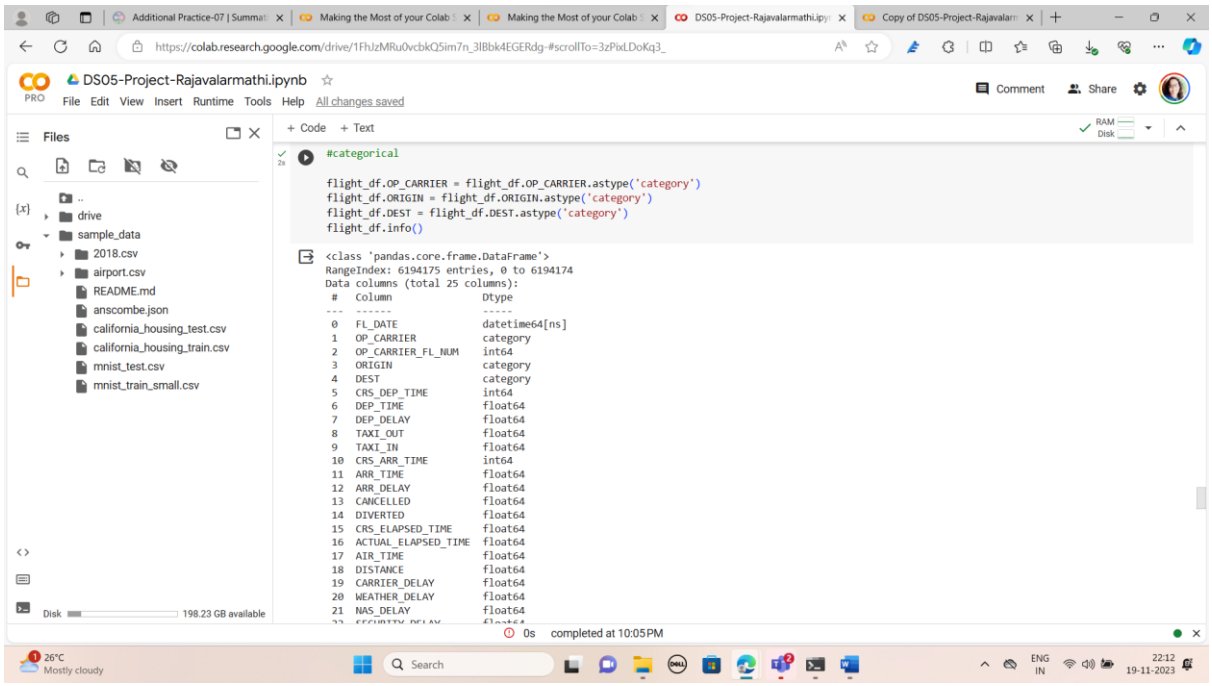


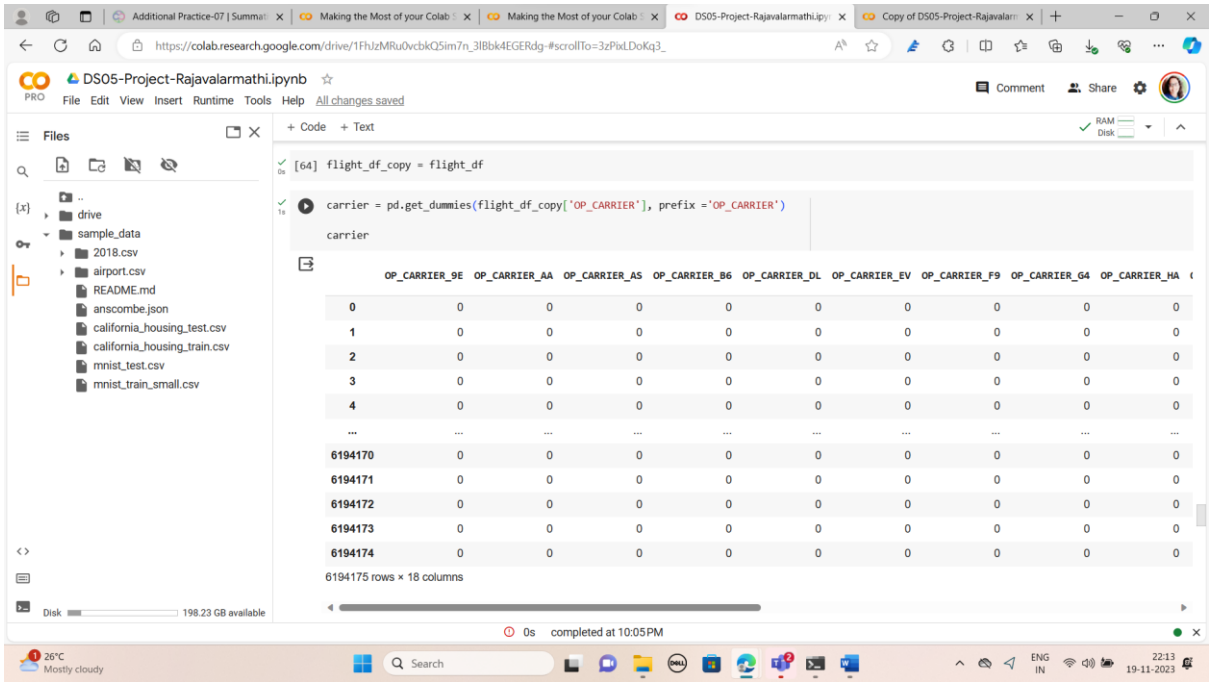



DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
sns.set_context("notebook")
snsg= sns.countplot(x = "OP_CARRIER", data = flight_df)
snsg.set(xlabel = "number of columns")
[Text(0.5, 0, 'number of columns')]
count
number of columns
United Airlines B6 EV F9 G4 HA MQ NK OH OO VX WN YV YX AA DL
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
0s completed at 10:05PM


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 29 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER object
2 OP_CARRIER_FL_NUM int64
3 ORIGIN object
4 DEST object
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 WHEELS_OFF float64
10 WHEELS_ON float64
11 TAXI_IN float64
12 CRS_ARR_TIME int64
13 ARR_TIME float64
14 ARR_DELAY float64
15 CANCELLED float64
16 CANCELLATION_CODE object
17 DIVERTED float64
18 CRS_ELAPSED_TIME float64
19 ACTUAL_ELAPSED_TIME float64
20 AIR_TIME float64
21 DISTANCE float64
22 CARRIER_DELAY float64
23 WEATHER_DELAY float64
24 NAS_DELAY float64
25 SECURITY_DELAY float64
26 LATE_AIRCRAFT_DELAY float64
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
0s completed at 10:05PM


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
+ Code + Text
flight_df.drop(['Unnamed: 27', 'CANCELLATION_CODE',
'WHEELS_OFF', 'WHEELS_ON'], axis =1, inplace= True)
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 25 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER object
2 OP_CARRIER_FL_NUM int64
3 ORIGIN object
4 DEST object
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 TAXI_IN float64
10 CRS_ARR_TIME int64
11 ARR_TIME float64
12 ARR_DELAY float64
13 CANCELLED float64
14 DIVERTED float64
15 CRS_ELAPSED_TIME float64
16 ACTUAL_ELAPSED_TIME float64
17 AIR_TIME float64
18 DISTANCE float64
19 CARRIER_DELAY float64
20 WEATHER_DELAY float64
21 NAS_DELAY float64
22 SECURITY_DELAY float64
23 LATE_AIRCRAFT_DELAY float64
24 flight status int64
0s completed at 10:05PM

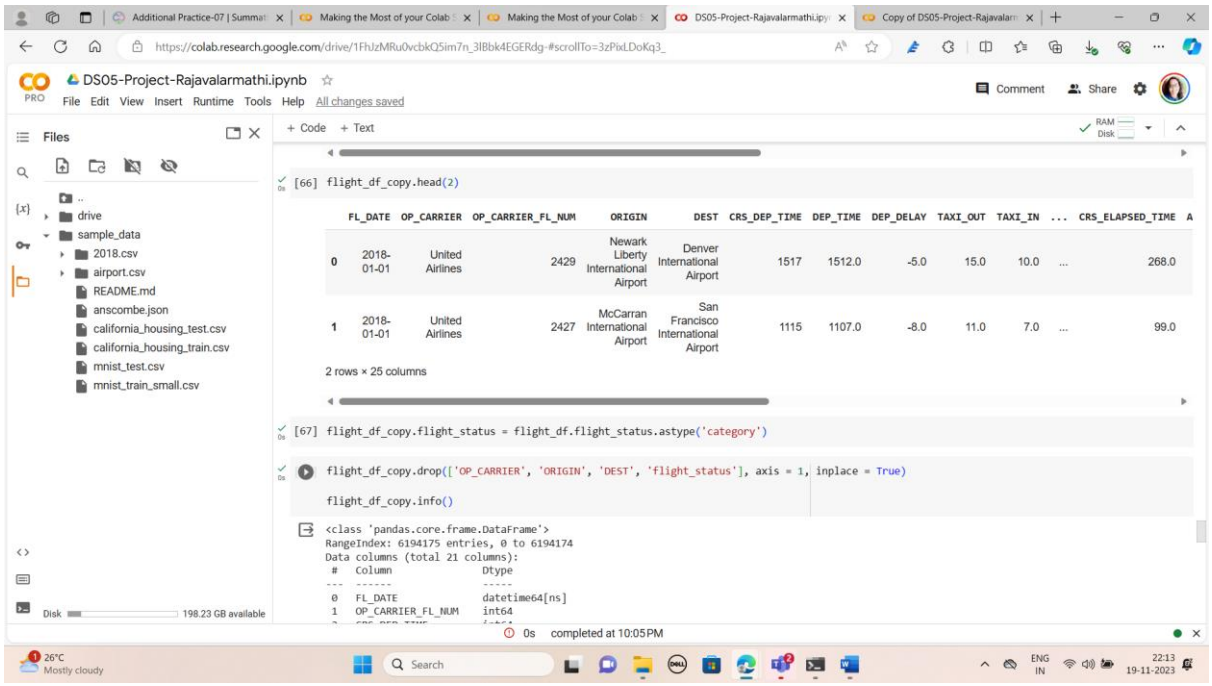

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
+ Code + Text
#categorical
flight_df.OP_CARRIER = flight_df.OP_CARRIER.astype('category')
flight_df.ORIGIN = flight_df.ORIGIN.astype('category')
flight_df.DEST = flight_df.DEST.astype('category')
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 25 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER category
2 OP_CARRIER_FL_NUM int64
3 ORIGIN category
4 DEST category
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 TAXI_IN float64
10 CRS_ARR_TIME int64
11 ARR_TIME float64
12 ARR_DELAY float64
13 CANCELLED float64
14 DIVERTED float64
15 CRS_ELAPSED_TIME float64
16 ACTUAL_ELAPSED_TIME float64
17 AIR_TIME float64
18 DISTANCE float64
19 CARRIER_DELAY float64
20 WEATHER_DELAY float64
21 NAS_DELAY float64
0s completed at 10:05PM



```
[64] flight_df_copy = flight_df

carrier = pd.get_dummies(flight_df_copy['OP_CARRIER'], prefix ='OP_CARRIER')

carrier
```

	OP_CARRIER_9E	OP_CARRIER_AA	OP_CARRIER_AS	OP_CARRIER_B6	OP_CARRIER_DL	OP_CARRIER_EV	OP_CARRIER_F9	OP_CARRIER_G4	OP_CARRIER_HA
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
6194170	0	0	0	0	0	0	0	0	0
6194171	0	0	0	0	0	0	0	0	0
6194172	0	0	0	0	0	0	0	0	0
6194173	0	0	0	0	0	0	0	0	0
6194174	0	0	0	0	0	0	0	0	0

6194175 rows × 18 columns



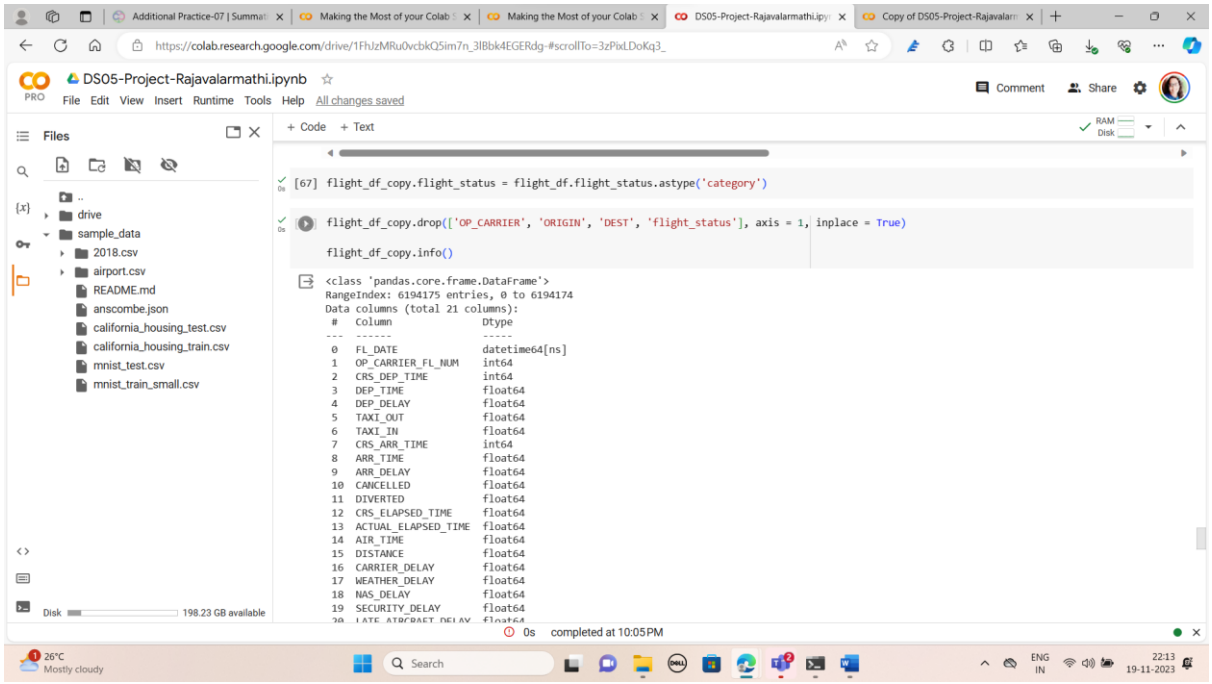
```
[66] flight_df_copy.head(2)
```

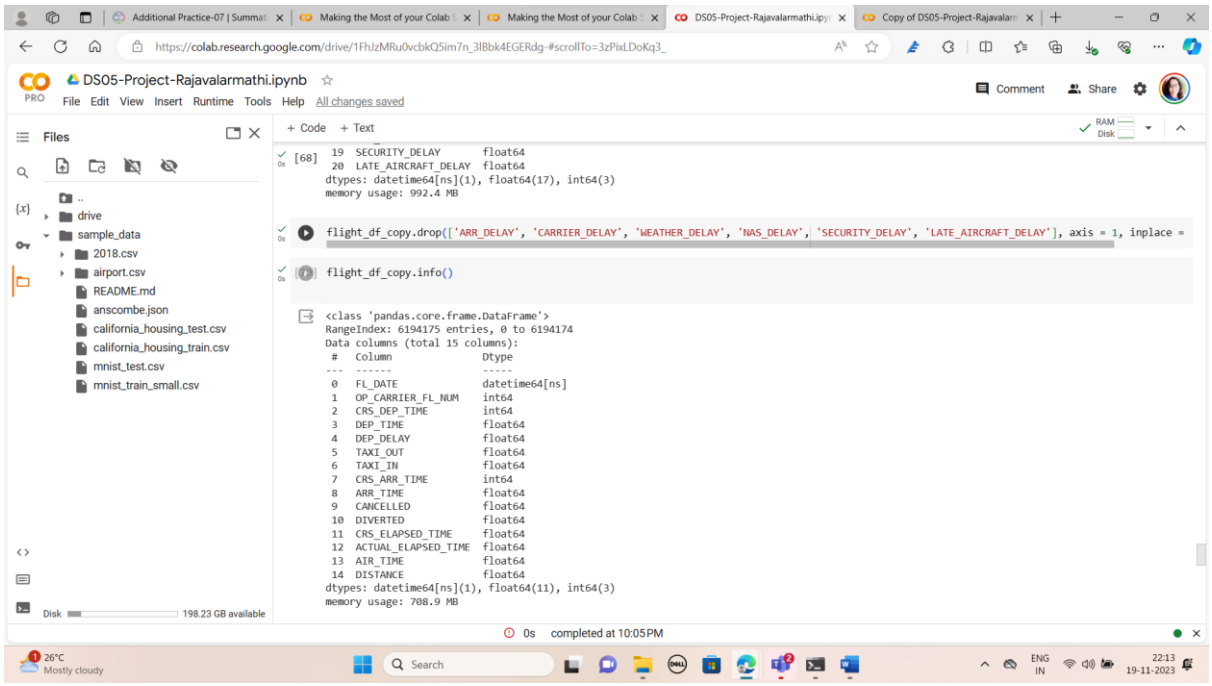
	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY	TAXI_OUT	TAXI_IN	...	CRS_ELAPSED_TIME	A
0	2018-01-01	United Airlines	2429	Newark Liberty International Airport	Denver International Airport	1517	1512.0	-5.0	15.0	10.0	...	268.0	
1	2018-01-01	United Airlines	2427	McCarran International Airport	San Francisco International Airport	1115	1107.0	-8.0	11.0	7.0	...	99.0	

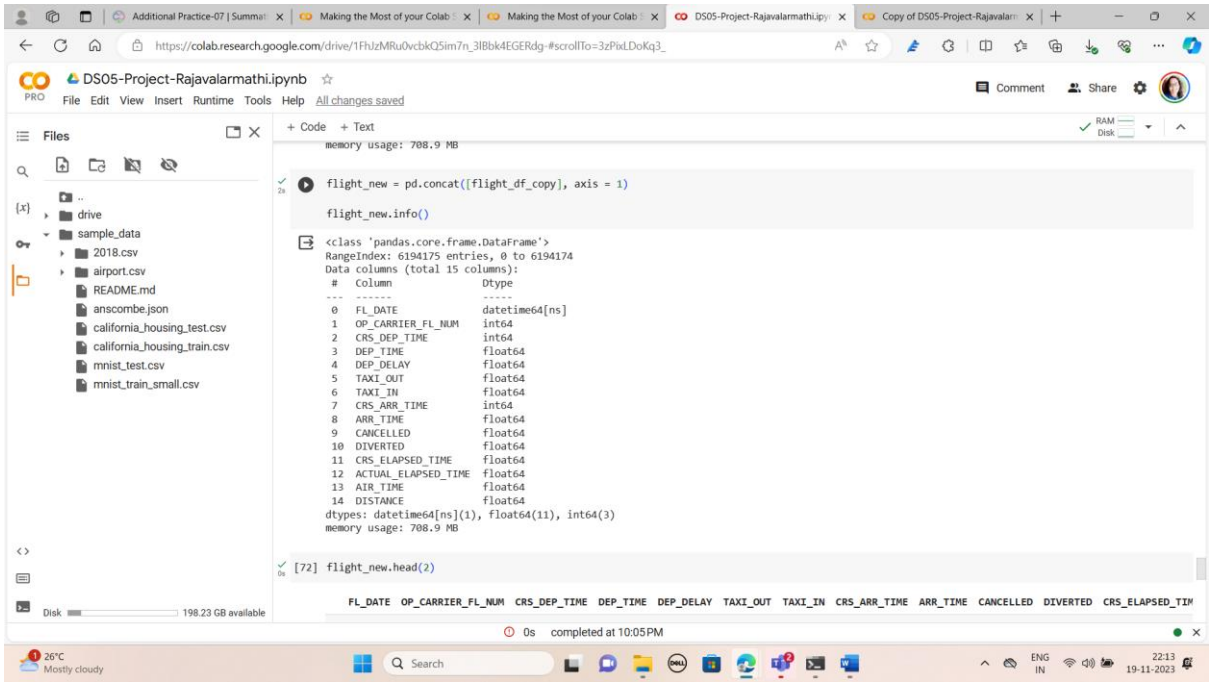
2 rows × 25 columns

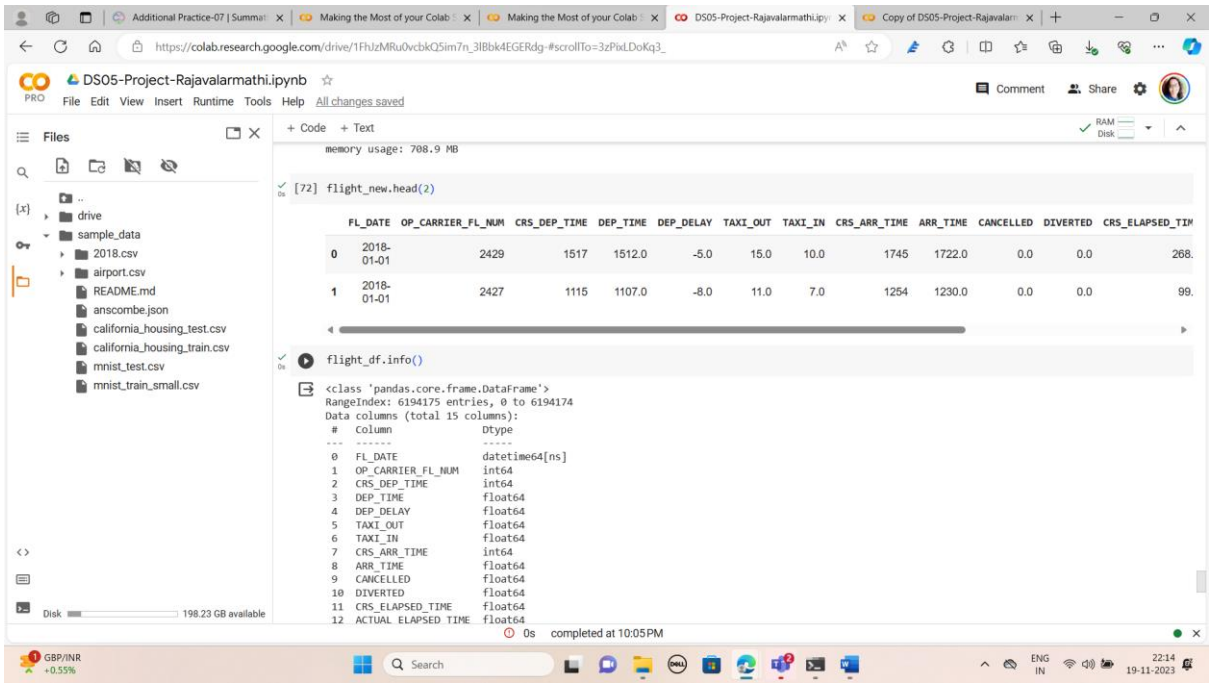
```
[67] flight_df_copy.flight_status = flight_df.flight_status.astype('category')

flight_df_copy.drop(['OP_CARRIER', 'ORIGIN', 'DEST', 'flight_status'], axis = 1, inplace = True)

flight_df_copy.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 21 columns):
 #   Column              Dtype
---  ------              -----
 0   FL_DATE             datetime64[ns]
 1   OP_CARRIER_FL_NUM   int64
```

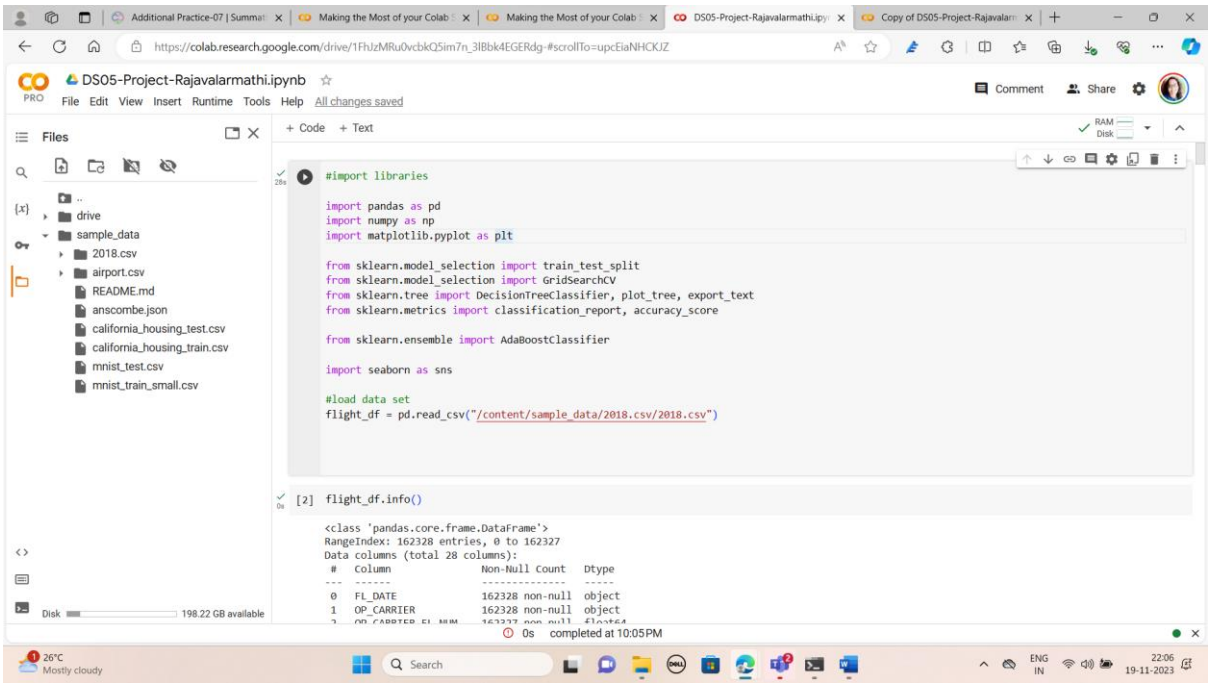

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
+ Code + Text
[67] flight_df_copy.flight_status = flight_df.flight_status.astype('category')
flight_df_copy.drop(['OP_CARRIER', 'ORIGIN', 'DEST', 'flight_status'], axis = 1, inplace = True)
flight_df_copy.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 21 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER_FL_NUM int64
2 CRS_DEP_TIME int64
3 DEP_TIME float64
4 DEP_DELAY float64
5 TAXI_OUT float64
6 TAXI_IN float64
7 CRS_ARR_TIME int64
8 ARR_TIME float64
9 ARR_DELAY float64
10 CANCELLED float64
11 DIVERTED float64
12 CRS_ELAPSED_TIME float64
13 ACTUAL_ELAPSED_TIME float64
14 AIR_TIME float64
15 DISTANCE float64
16 CARRIER_DELAY float64
17 WEATHER_DELAY float64
18 NAS_DELAY float64
19 SECURITY_DELAY float64
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:13 19-11-2023


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
+ Code + Text
[68] 19 SECURITY_DELAY float64
20 LATE_AIRCRAFT_DELAY float64
dtypes: datetime64[ns](1), float64(17), int64(3)
memory usage: 992.4 MB
flight_df_copy.drop(['ARR_DELAY', 'CARRIER_DELAY', 'WEATHER_DELAY', 'NAS_DELAY', 'SECURITY_DELAY', 'LATE_AIRCRAFT_DELAY'], axis = 1, inplace =
flight_df_copy.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 15 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER_FL_NUM int64
2 CRS_DEP_TIME int64
3 DEP_TIME float64
4 DEP_DELAY float64
5 TAXI_OUT float64
6 TAXI_IN float64
7 CRS_ARR_TIME int64
8 ARR_TIME float64
9 CANCELLED float64
10 DIVERTED float64
11 CRS_ELAPSED_TIME float64
12 ACTUAL_ELAPSED_TIME float64
13 AIR_TIME float64
14 DISTANCE float64
dtypes: datetime64[ns](1), float64(11), int64(3)
memory usage: 708.9 MB
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:13 19-11-2023

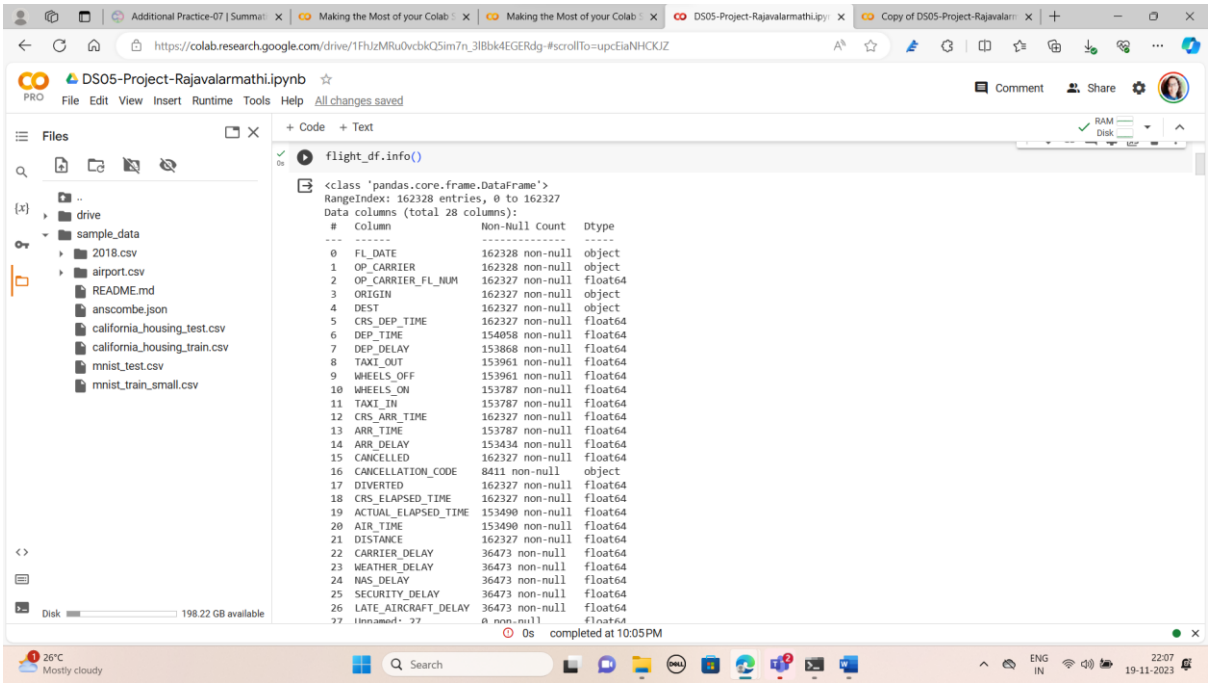

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
memory usage: 708.9 MB
flight_new = pd.concat([flight_df_copy], axis = 1)
flight_new.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 15 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER_FL_NUM int64
2 CRS_DEP_TIME int64
3 DEP_TIME float64
4 DEP_DELAY float64
5 TAXI_OUT float64
6 TAXI_IN float64
7 CRS_ARR_TIME int64
8 ARR_TIME float64
9 CANCELLED float64
10 DIVERTED float64
11 CRS_ELAPSED_TIME float64
12 ACTUAL_ELAPSED_TIME float64
13 AIR_TIME float64
14 DISTANCE float64
dtypes: datetime64[ns](1), float64(11), int64(3)
memory usage: 708.9 MB
[72] flight_new.head(2)
FL_DATE OP_CARRIER_FL_NUM CRS_DEP_TIME DEP_TIME DEP_DELAY TAXI_OUT TAXI_IN CRS_ARR_TIME ARR_TIME CANCELLED DIVERTED CRS_ELAPSED_TIM
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:13 19-11-2023

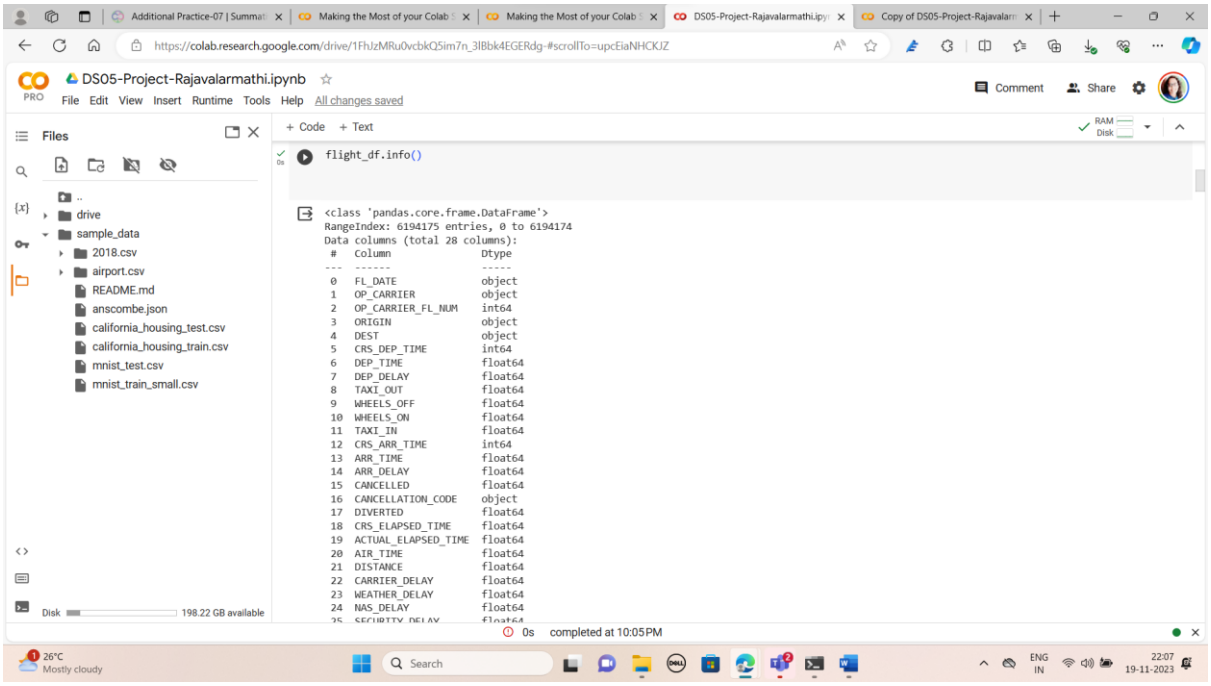

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.23 GB available
memory usage: 708.9 MB
[72] flight_new.head(2)
FL_DATE OP_CARRIER_FL_NUM CRS_DEP_TIME DEP_TIME DEP_DELAY TAXI_OUT TAXI_IN CRS_ARR_TIME ARR_TIME CANCELLED DIVERTED CRS_ELAPSED_TIM
0 2018-01-01 2429 1517 1512.0 -5.0 15.0 10.0 1745 1722.0 0.0 0.0 268.
1 2018-01-01 2427 1115 1107.0 -8.0 11.0 7.0 1254 1230.0 0.0 0.0 99.
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 15 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER_FL_NUM int64
2 CRS_DEP_TIME int64
3 DEP_TIME float64
4 DEP_DELAY float64
5 TAXI_OUT float64
6 TAXI_IN float64
7 CRS_ARR_TIME int64
8 ARR_TIME float64
9 CANCELLED float64
10 DIVERTED float64
11 CRS_ELAPSED_TIME float64
12 ACTUAL_ELAPSED_TIME float64
0s completed at 10:05PM
GBP/INR +0.55%
Search
ENG IN
22:14 19-11-2023

6.

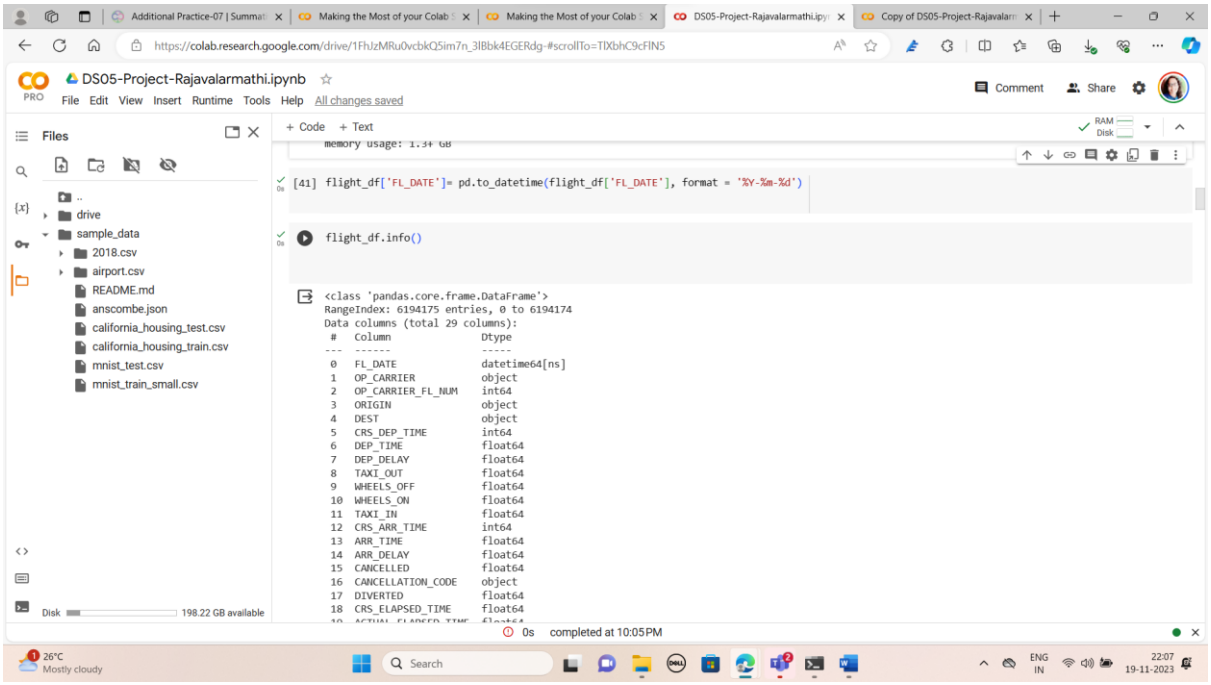

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
#import libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.tree import DecisionTreeClassifier, plot_tree, export_text
from sklearn.metrics import classification_report, accuracy_score
from sklearn.ensemble import AdaBoostClassifier
import seaborn as sns
#load data set
flight_df = pd.read_csv("/content/sample_data/2018.csv/2018.csv")
[2] flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162328 entries, 0 to 162327
Data columns (total 28 columns):
Column Non-Null Count Dtype
0 FL_DATE 162328 non-null object
1 OP_CARRIER 162328 non-null object
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
0s completed at 10:05PM

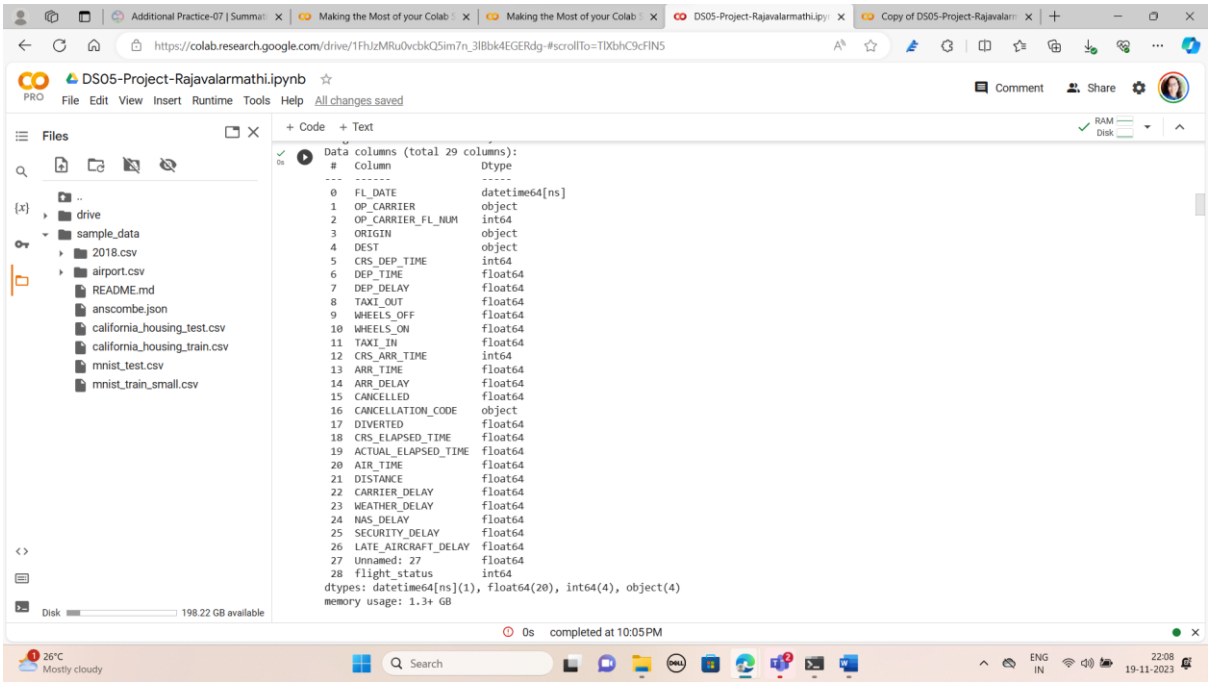
7.


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 162328 entries, 0 to 162327
Data columns (total 28 columns):
Column Non-Null Count Dtype
0 FL_DATE 162328 non-null object
1 OP_CARRIER 162328 non-null object
2 OP_CARRIER_FL_NUM 162327 non-null float64
3 ORIGIN 162327 non-null object
4 DEST 162327 non-null object
5 CRS_DEP_TIME 162327 non-null float64
6 DEP_TIME 154058 non-null float64
7 DEP_DELAY 153868 non-null float64
8 TAXI_OUT 153961 non-null float64
9 WHEELS_OFF 153961 non-null float64
10 WHEELS_ON 153787 non-null float64
11 TAXI_IN 153787 non-null float64
12 CRS_ARR_TIME 162327 non-null float64
13 ARR_TIME 153787 non-null float64
14 ARR_DELAY 153434 non-null float64
15 CANCELLED 162327 non-null float64
16 CANCELLATION_CODE 8411 non-null object
17 DIVERTED 162327 non-null float64
18 CRS_ELAPSED_TIME 162327 non-null float64
19 ACTUAL_ELAPSED_TIME 153490 non-null float64
20 AIR_TIME 153490 non-null float64
21 DISTANCE 162327 non-null float64
22 CARRIER_DELAY 36473 non-null float64
23 WEATHER_DELAY 36473 non-null float64
24 NAS_DELAY 36473 non-null float64
25 SECURITY_DELAY 36473 non-null float64
26 LATE_AIRCRAFT_DELAY 36473 non-null float64
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
0s completed at 10:05PM

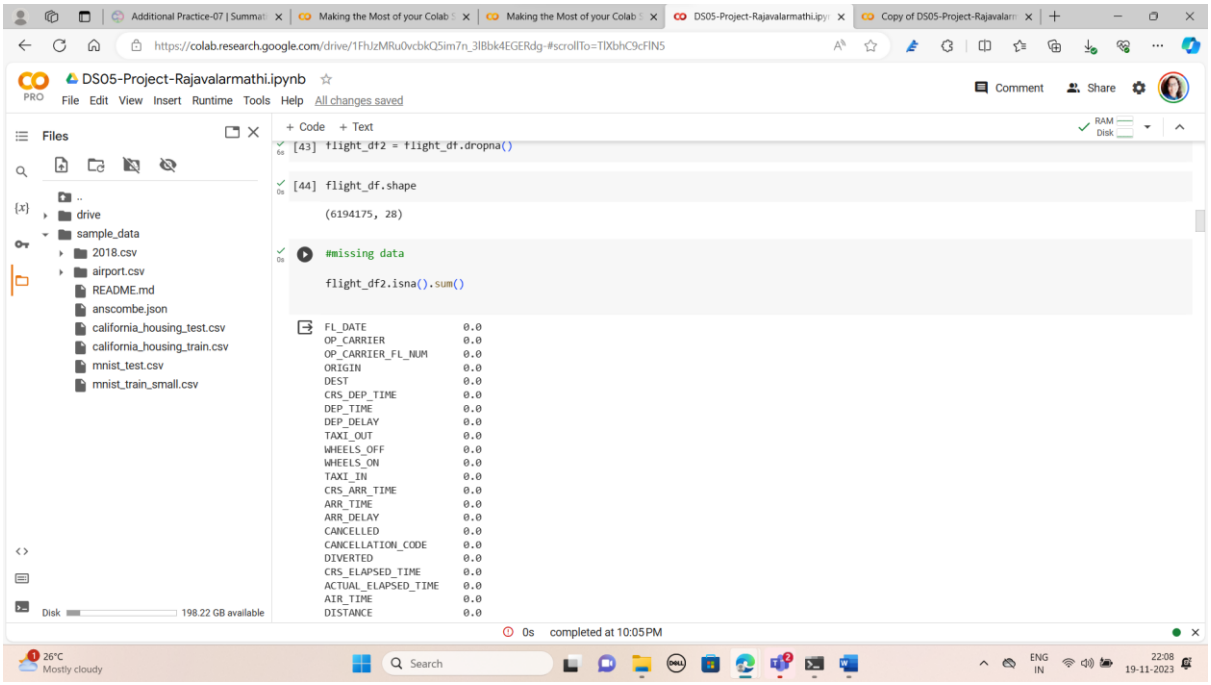

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 28 columns):
Column Dtype
0 FL_DATE object
1 OP_CARRIER object
2 OP_CARRIER_FL_NUM int64
3 ORIGIN object
4 DEST object
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 WHEELS_OFF float64
10 WHEELS_ON float64
11 TAXI_IN float64
12 CRS_ARR_TIME int64
13 ARR_TIME float64
14 ARR_DELAY float64
15 CANCELLED float64
16 CANCELLATION_CODE object
17 DIVERTED float64
18 CRS_ELAPSED_TIME float64
19 ACTUAL_ELAPSED_TIME float64
20 AIR_TIME float64
21 DISTANCE float64
22 CARRIER_DELAY float64
23 WEATHER_DELAY float64
24 NAS_DELAY float64
Disk 198.22 GB available
0s completed at 10:05PM

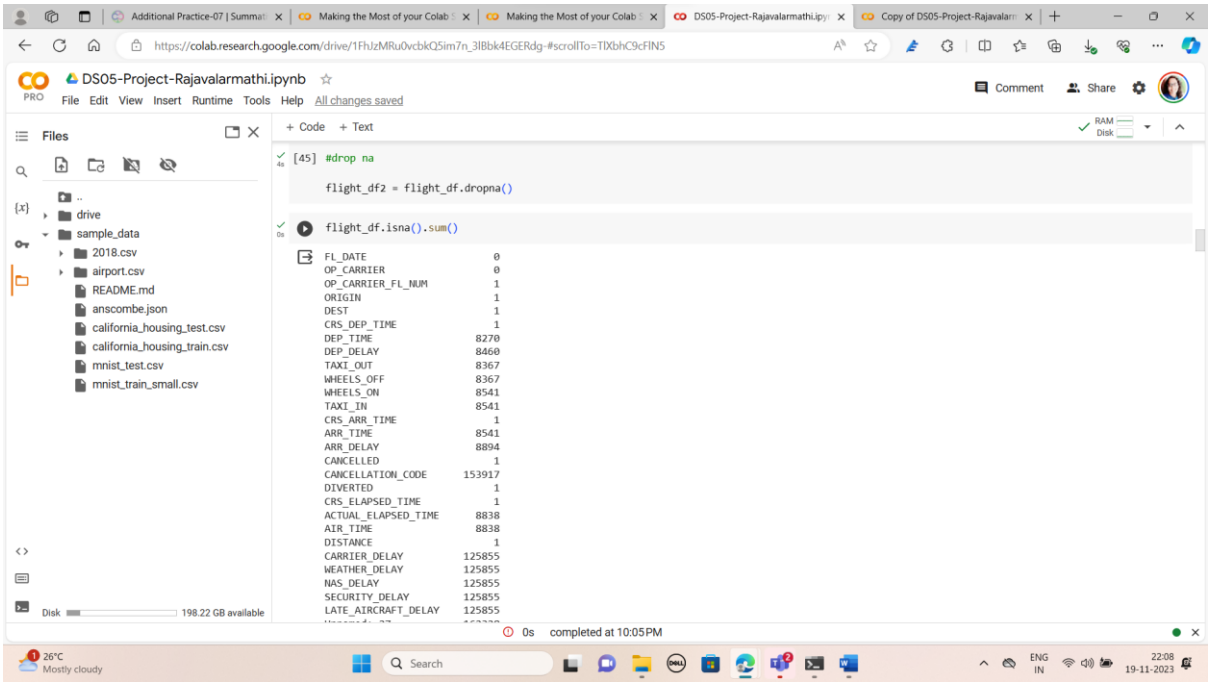
8.

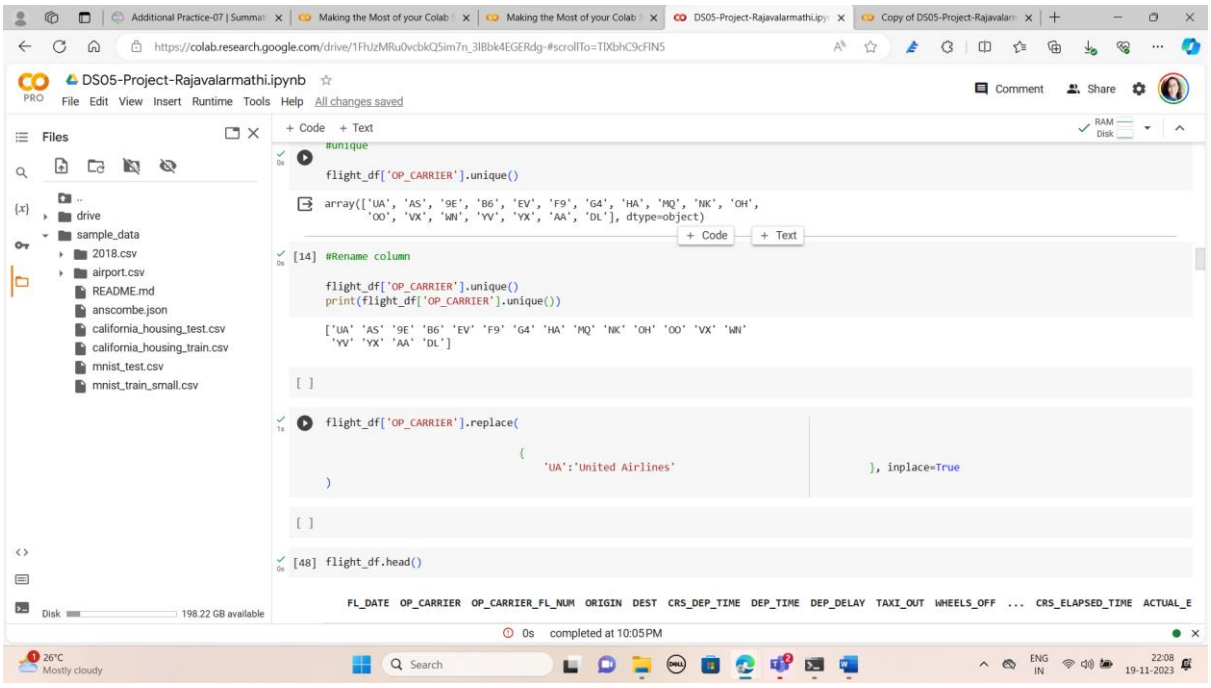

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
[41] flight_df['FL_DATE']= pd.to_datetime(flight_df['FL_DATE'], format = '%Y-%m-%d')
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 29 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER object
2 OP_CARRIER_FL_NUM int64
3 ORIGIN object
4 DEST object
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 WHEELS_OFF float64
10 WHEELS_ON float64
11 TAXI_IN float64
12 CRS_ARR_TIME int64
13 ARR_TIME float64
14 ARR_DELAY float64
15 CANCELLED float64
16 CANCELLATION_CODE object
17 DIVERTED float64
18 CRS_ELAPSED_TIME float64
Disk 198.22 GB available
0s completed at 10:05PM

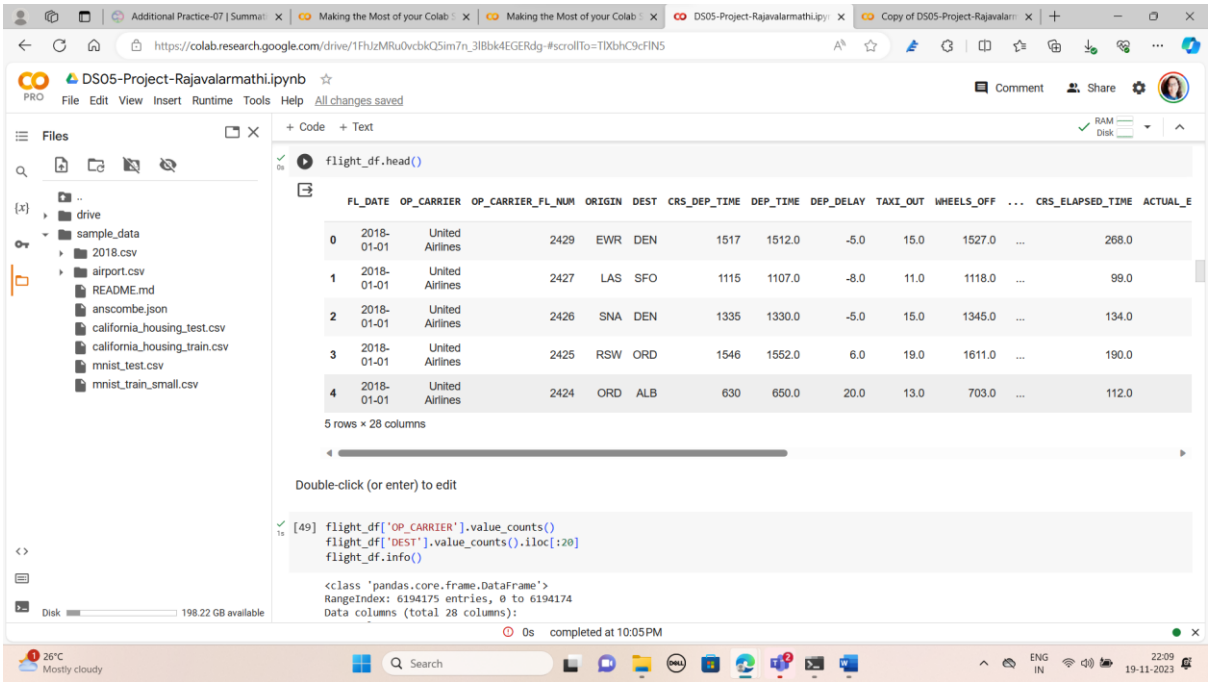

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Data columns (total 29 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER object
2 OP_CARRIER_FL_NUM int64
3 ORIGIN object
4 DEST object
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 WHEELS_OFF float64
10 WHEELS_ON float64
11 TAXI_IN float64
12 CRS_ARR_TIME int64
13 ARR_TIME float64
14 ARR_DELAY float64
15 CANCELLED float64
16 CANCELLATION_CODE object
17 DIVERTED float64
18 CRS_ELAPSED_TIME float64
19 ACTUAL_ELAPSED_TIME float64
20 AIR_TIME float64
21 DISTANCE float64
22 CARRIER_DELAY float64
23 WEATHER_DELAY float64
24 NAS_DELAY float64
25 SECURITY_DELAY float64
26 LATE_AIRCRAFT_DELAY float64
27 Unnamed: 27 float64
28 flight_status int64
dtypes: datetime64[ns](1), float64(20), int64(4), object(4)
memory usage: 1.3+ GB
Disk 198.22 GB available
0s completed at 10:05PM

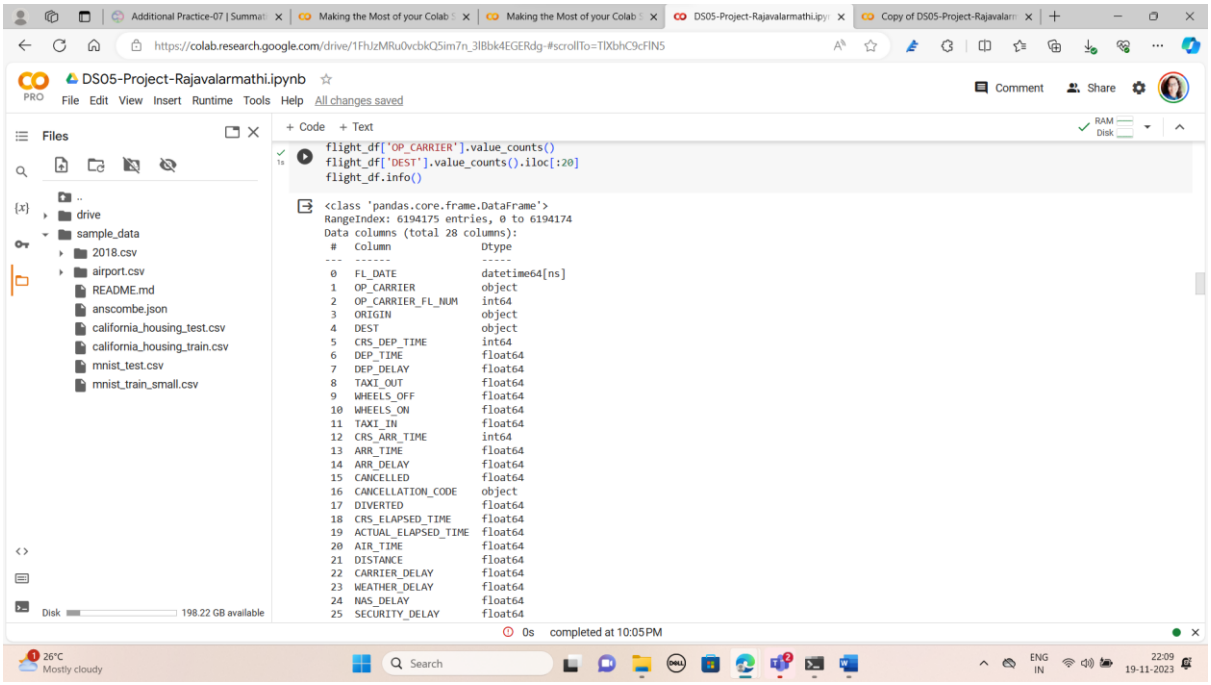
9.

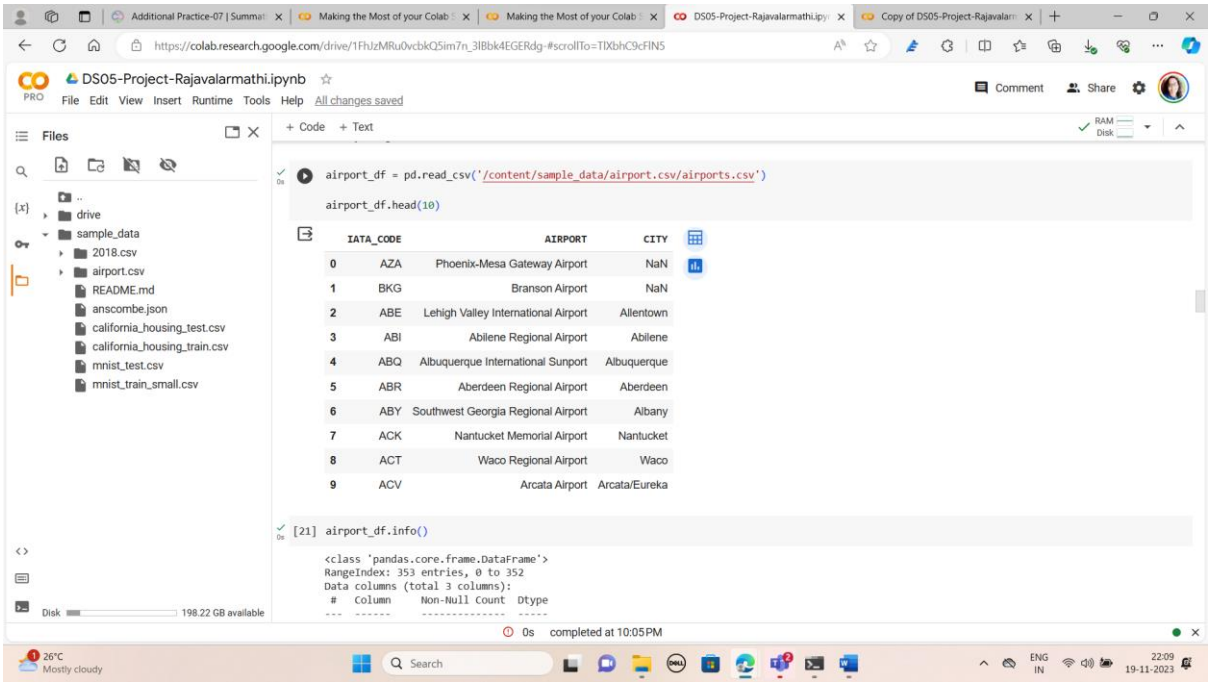

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
[43] flight_df2 = flight_df.dropna()
[44] flight_df.shape
(6194175, 28)
#missing data
flight_df2.isna().sum()
FL_DATE 0.0
OP_CARRIER 0.0
OP_CARRIER_FL_NUM 0.0
ORIGIN 0.0
DEST 0.0
CRS_DEP_TIME 0.0
DEP_TIME 0.0
DEP_DELAY 0.0
TAXI_OUT 0.0
WHEELS_OFF 0.0
WHEELS_ON 0.0
TAXI_IN 0.0
CRS_ARR_TIME 0.0
ARR_TIME 0.0
ARR_DELAY 0.0
CANCELLED 0.0
CANCELLATION_CODE 0.0
DIVERTED 0.0
CRS_ELAPSED_TIME 0.0
ACTUAL_ELAPSED_TIME 0.0
AIR_TIME 0.0
DISTANCE 0.0
Disk 198.22 GB available
0s completed at 10:05PM

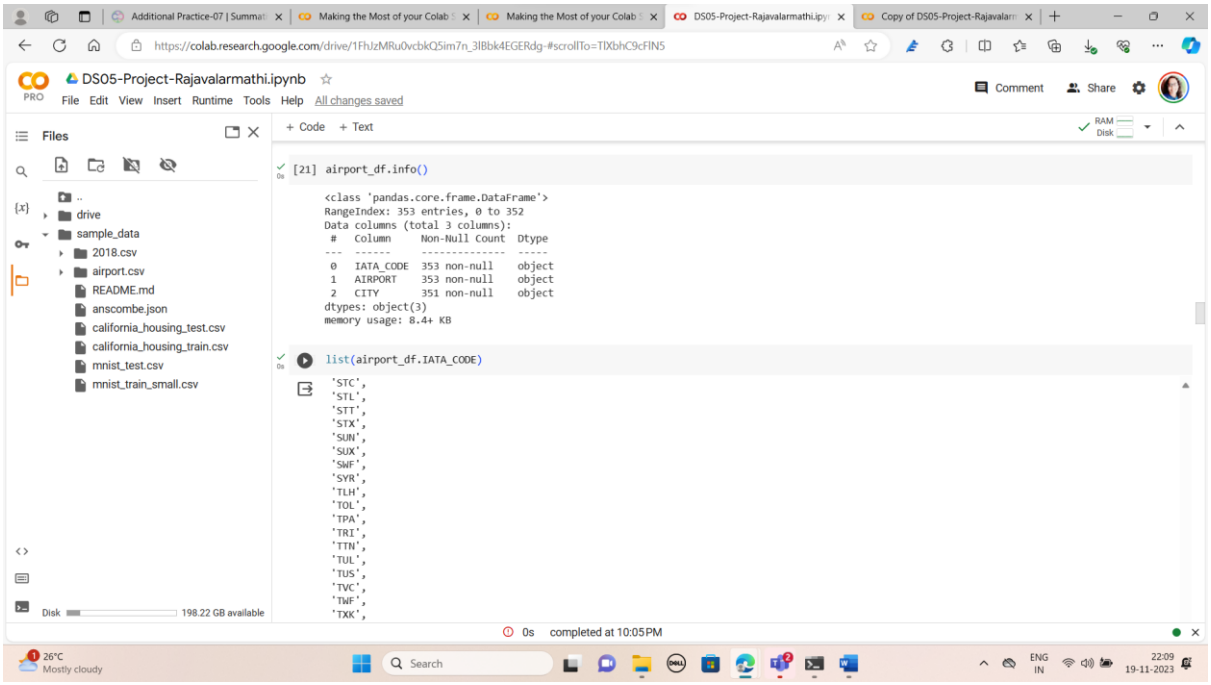

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
[45] #drop na
flight_df2 = flight_df.dropna()
flight_df.isna().sum()
FL_DATE 0
OP_CARRIER 0
OP_CARRIER_FL_NUM 1
ORIGIN 1
DEST 1
CRS_DEP_TIME 1
DEP_TIME 8270
DEP_DELAY 8460
TAXI_OUT 8367
WHEELS_OFF 8367
WHEELS_ON 8541
TAXI_IN 8541
CRS_ARR_TIME 1
ARR_TIME 8541
ARR_DELAY 8894
CANCELLED 1
CANCELLATION_CODE 153917
DIVERTED 1
CRS_ELAPSED_TIME 1
ACTUAL_ELAPSED_TIME 8838
AIR_TIME 8838
DISTANCE 1
CARRIER_DELAY 125855
WEATHER_DELAY 125855
NAS_DELAY 125855
SECURITY_DELAY 125855
LATE_AIRCRAFT_DELAY 125855
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
0s completed at 10:05PM

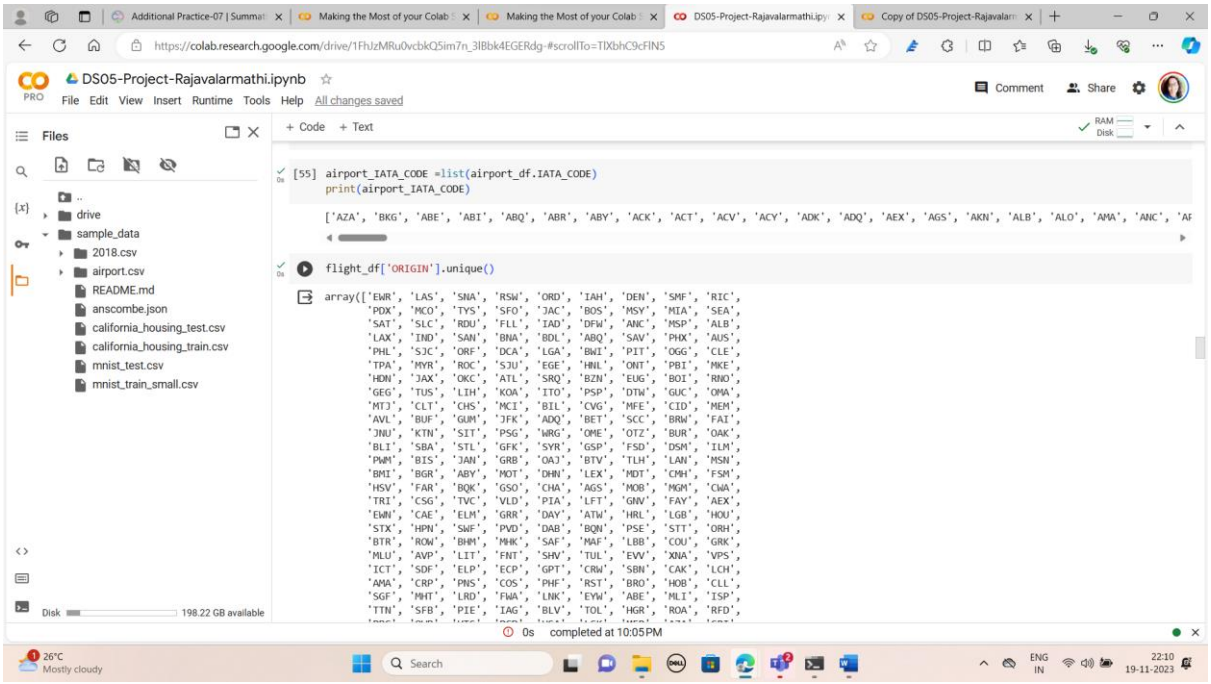

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
#unique
flight_df['OP_CARRIER'].unique()
array(['UA', 'AS', '9E', 'B6', 'EV', 'F9', 'G4', 'HA', 'MQ', 'NK', 'OH',
'OO', 'VX', 'WN', 'YV', 'YX', 'AA', 'DL'], dtype=object)
+ Code + Text
[14] #Rename column
flight_df['OP_CARRIER'].unique()
print(flight_df['OP_CARRIER'].unique())
['UA' 'AS' '9E' 'B6' 'EV' 'F9' 'G4' 'HA' 'MQ' 'NK' 'OH' 'OO' 'VX' 'WN'
'YV' 'YX' 'AA' 'DL']
[ ]
flight_df['OP_CARRIER'].replace(
{
'UA':'United Airlines'
}, inplace=True
)
[ ]
[48] flight_df.head()
FL_DATE OP_CARRIER OP_CARRIER_FL_NUM ORIGIN DEST CRS_DEP_TIME DEP_TIME DEP_DELAY TAXI_OUT WHEELS_OFF ... CRS_ELAPSED_TIME ACTUAL_E
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
0s completed at 10:05PM

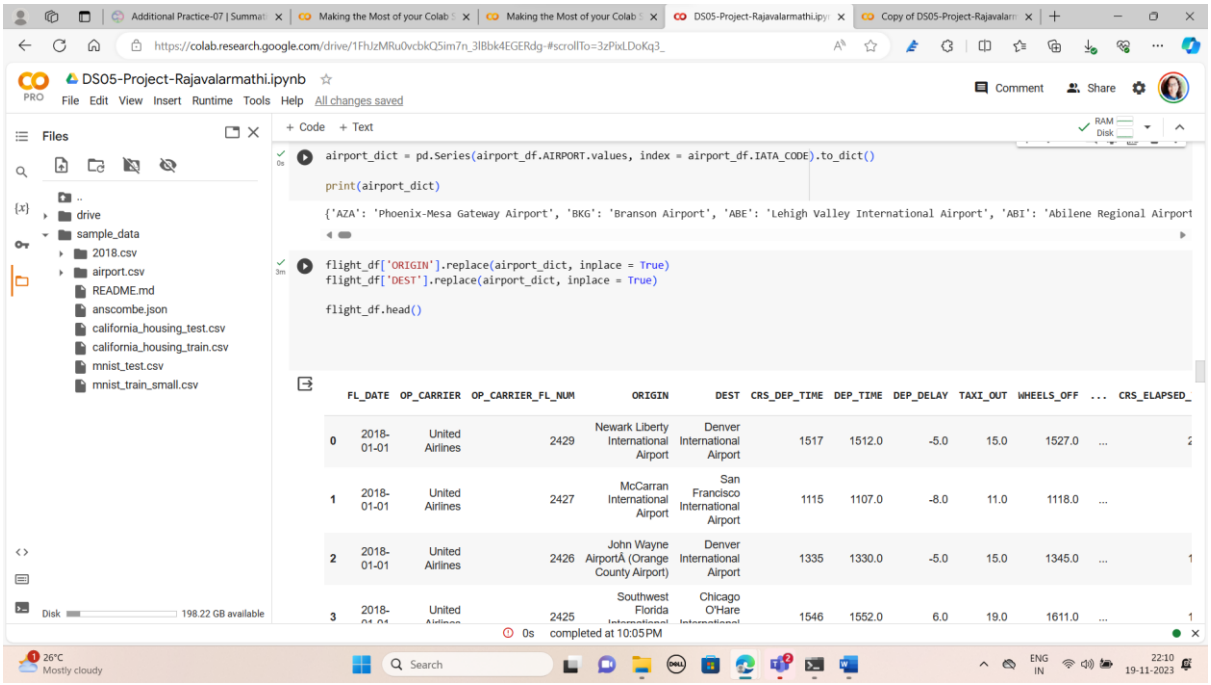

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
flight_df.head()
FL_DATE OP_CARRIER OP_CARRIER_FL_NUM ORIGIN DEST CRS_DEP_TIME DEP_TIME DEP_DELAY TAXI_OUT WHEELS_OFF ... CRS_ELAPSED_TIME ACTUAL_E
0 2018-01-01 United Airlines 2429 EWR DEN 1517 1512.0 -5.0 15.0 1527.0 ... 268.0
1 2018-01-01 United Airlines 2427 LAS SFO 1115 1107.0 -8.0 11.0 1118.0 ... 99.0
2 2018-01-01 United Airlines 2426 SNA DEN 1335 1330.0 -5.0 15.0 1345.0 ... 134.0
3 2018-01-01 United Airlines 2425 RSW ORD 1546 1552.0 6.0 19.0 1611.0 ... 190.0
4 2018-01-01 United Airlines 2424 ORD ALB 630 650.0 20.0 13.0 703.0 ... 112.0
5 rows × 28 columns
Double-click (or enter) to edit
[49] flight_df['OP_CARRIER'].value_counts()
flight_df['DEST'].value_counts().iloc[:20]
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 28 columns):
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:09 19-11-2023

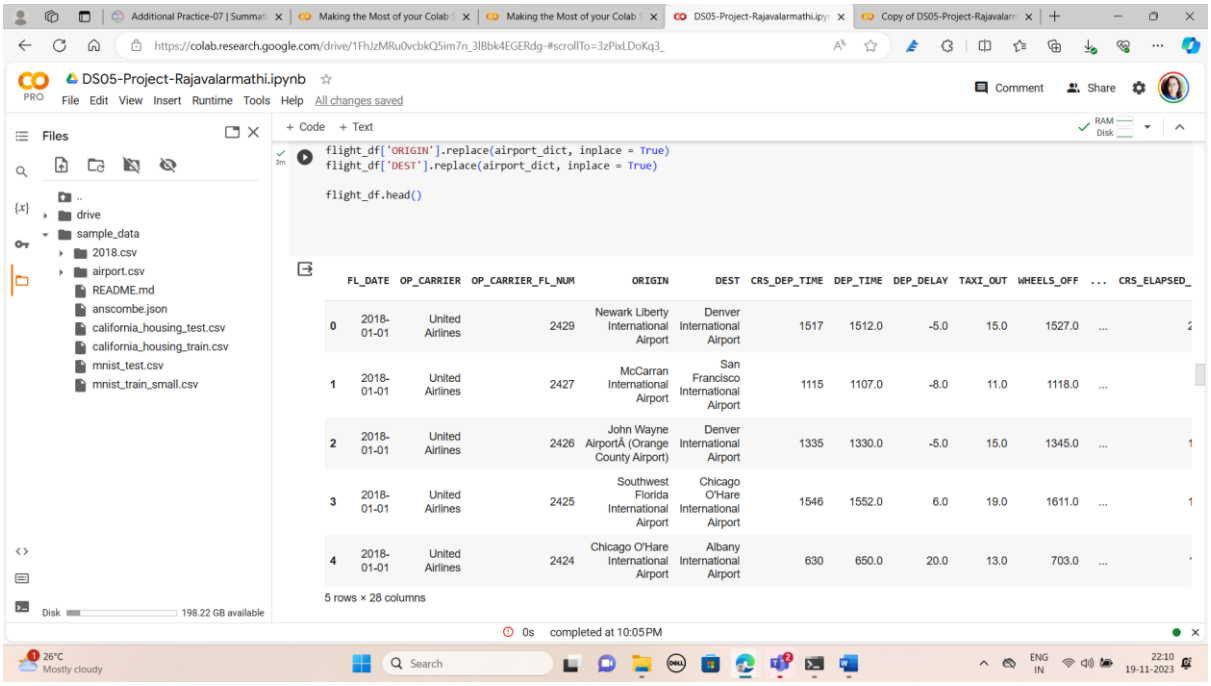

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
flight_df['OP_CARRIER'].value_counts()
flight_df['DEST'].value_counts().iloc[:20]
flight_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6194175 entries, 0 to 6194174
Data columns (total 28 columns):
Column Dtype
0 FL_DATE datetime64[ns]
1 OP_CARRIER object
2 OP_CARRIER_FL_NUM int64
3 ORIGIN object
4 DEST object
5 CRS_DEP_TIME int64
6 DEP_TIME float64
7 DEP_DELAY float64
8 TAXI_OUT float64
9 WHEELS_OFF float64
10 WHEELS_ON float64
11 TAXI_IN float64
12 CRS_ARR_TIME int64
13 ARR_TIME float64
14 ARR_DELAY float64
15 CANCELLED float64
16 CANCELLATION_CODE object
17 DIVERTED float64
18 CRS_ELAPSED_TIME float64
19 ACTUAL_ELAPSED_TIME float64
20 AIR_TIME float64
21 DISTANCE float64
22 CARRIER_DELAY float64
23 WEATHER_DELAY float64
24 NAS_DELAY float64
25 SECURITY_DELAY float64
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:09 19-11-2023


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
airport_df = pd.read_csv('/content/sample_data/airport.csv/airports.csv')
airport_df.head(10)
IATA_CODE AIRPORT CITY
0 AZA Phoenix-Mesa Gateway Airport NaN
1 BKG Branson Airport NaN
2 ABE Lehigh Valley International Airport Allentown
3 ABI Abilene Regional Airport Abilene
4 ABQ Albuquerque International Sunport Albuquerque
5 ABR Aberdeen Regional Airport Aberdeen
6 ABY Southwest Georgia Regional Airport Albany
7 ACK Nantucket Memorial Airport Nantucket
8 ACT Waco Regional Airport Waco
9 ACV Arcata Airport Arcata/Eureka
[21] airport_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 353 entries, 0 to 352
Data columns (total 3 columns):
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:09 19-11-2023

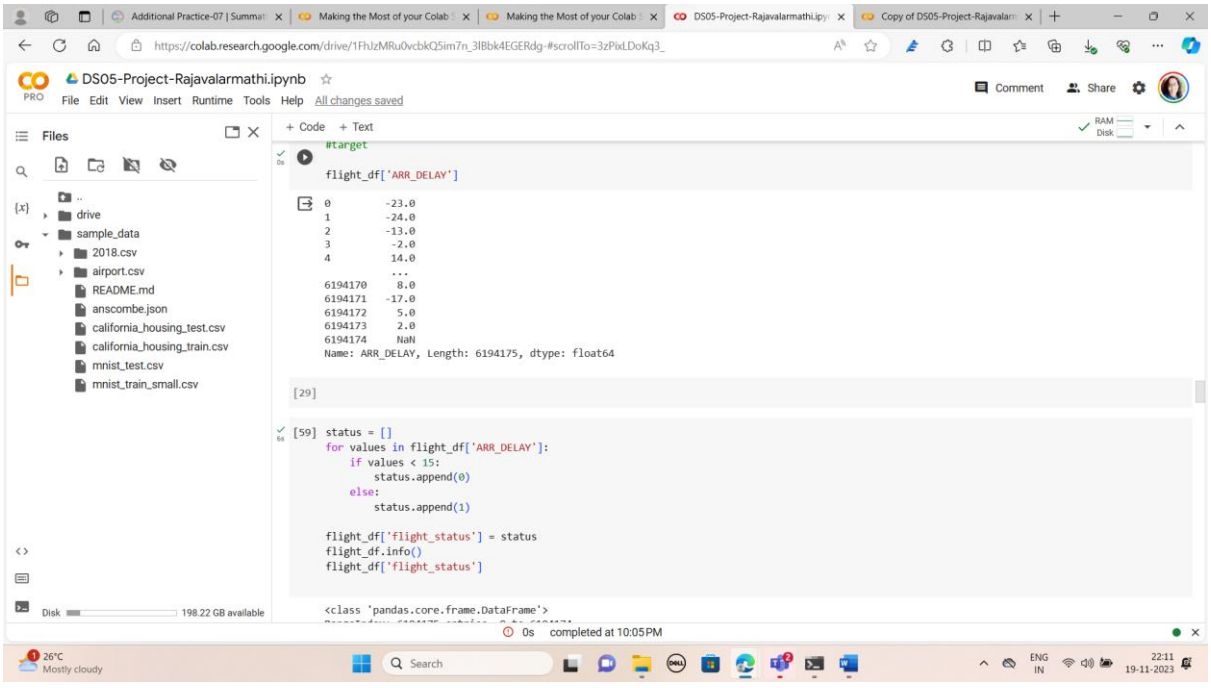

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
Files
+ Code + Text
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
[21] airport_df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 353 entries, 0 to 352
Data columns (total 3 columns):
Column Non-Null Count Dtype
0 IATA_CODE 353 non-null object
1 AIRPORT 353 non-null object
2 CITY 351 non-null object
dtypes: object(3)
memory usage: 8.4+ KB
list(airport_df.IATA_CODE)
'STC',
'STL',
'STT',
'STX',
'SUN',
'SUX',
'SWF',
'SYR',
'TLH',
'TOL',
'TPA',
'TRI',
'TTN',
'TUL',
'TUS',
'TVC',
'TWF',
'TXK',
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:09 19-11-2023

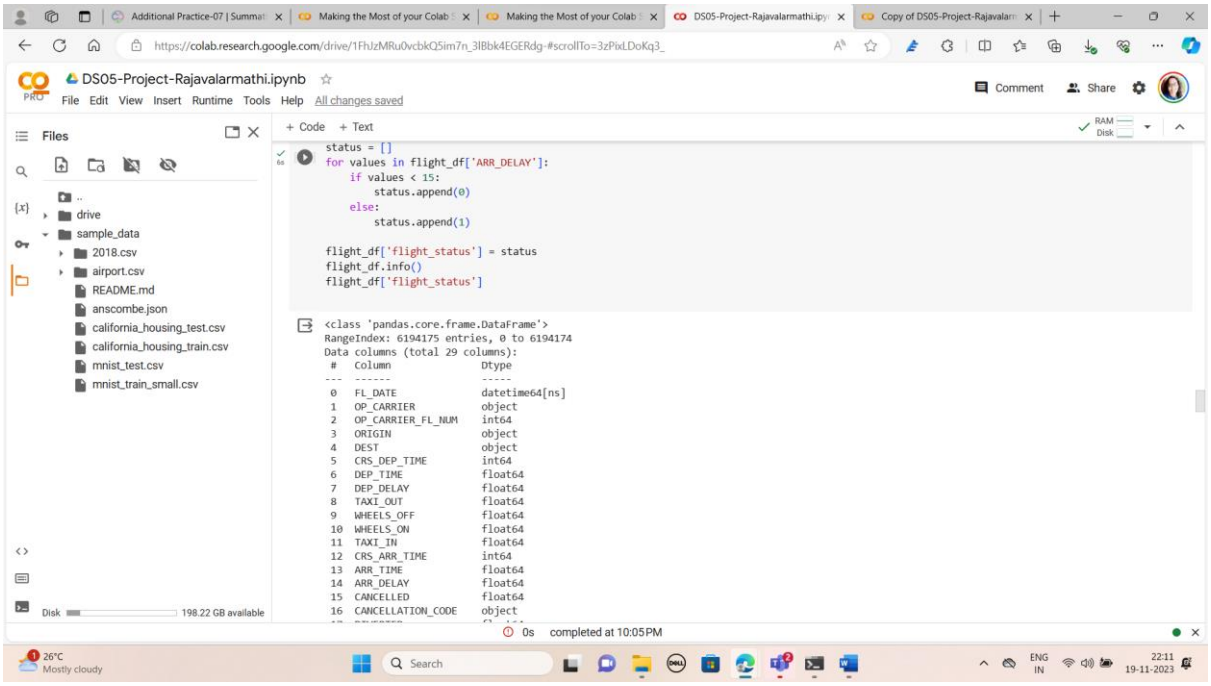

DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
[55] airport_IATA_CODE =list(airport_df.IATA_CODE)
print(airport_IATA_CODE)
['AZA', 'BKG', 'ABE', 'ABI', 'ABQ', 'ABR', 'ABY', 'ACK', 'ACT', 'ACV', 'ACY', 'ADK', 'ADQ', 'AEX', 'AGS', 'AKN', 'ALB', 'ALO', 'AMA', 'ANC', 'AF
flight_df['ORIGIN'].unique()
array(['EWR', 'LAS', 'SNA', 'RSW', 'ORD', 'IAH', 'DEN', 'SMF', 'RIC',
'PDX', 'MCO', 'TYS', 'SFO', 'JAC', 'BOS', 'MSY', 'MIA', 'SEA',
'SAT', 'SLC', 'RDU', 'FLL', 'IAD', 'DFW', 'ANC', 'MSP', 'ALB',
'LAX', 'IND', 'SAN', 'BNA', 'BDL', 'ABQ', 'SAV', 'PHX', 'AUS',
'PHL', 'SJC', 'ORF', 'DCA', 'LGA', 'BWI', 'PIT', 'OGG', 'CLE',
'TPA', 'MYR', 'ROC', 'SJU', 'EGE', 'HNL', 'ONT', 'PBI', 'MKE',
'HDN', 'JAX', 'OKC', 'ATL', 'SRQ', 'BZN', 'EUG', 'BOI', 'RNO',
'GEG', 'TUS', 'LIH', 'KOA', 'ITO', 'PSP', 'DTW', 'GUC', 'OMA',
'MTJ', 'CLT', 'CHS', 'MCI', 'BIL', 'CVG', 'MFE', 'CID', 'MEM',
'AVL', 'BUF', 'GUM', 'JFK', 'ADQ', 'BET', 'SCC', 'BRW', 'FAI',
'JNU', 'KTN', 'SIT', 'PSG', 'WRG', 'OME', 'OTZ', 'BUR', 'OAK',
'BLI', 'SBA', 'STL', 'GFK', 'SYR', 'GSP', 'FSD', 'DSM', 'ILM',
'PWM', 'BIS', 'JAN', 'GRB', 'OAJ', 'BTV', 'TLH', 'LAN', 'MSN',
'BMI', 'BGR', 'ABY', 'MOT', 'DHN', 'LEX', 'MDT', 'CMH', 'FSM',
'HSV', 'FAR', 'BQK', 'GSO', 'CHA', 'AGS', 'MOB', 'MGM', 'CWA',
'TRI', 'CSG', 'TVC', 'VLD', 'PIA', 'LFT', 'GNV', 'FAY', 'AEX',
'EWN', 'CAE', 'ELM', 'GRR', 'DAY', 'ATW', 'HRL', 'LGB', 'HOU',
'STX', 'HPN', 'SWF', 'PVD', 'DAB', 'BQN', 'PSE', 'STT', 'ORH',
'BTR', 'ROW', 'BHM', 'MHK', 'SAF', 'MAF', 'LBB', 'COU', 'GRK',
'MLU', 'AVP', 'LIT', 'FNT', 'SHV', 'TUL', 'EVV', 'XNA', 'VPS',
'ICT', 'SDF', 'ELP', 'ECP', 'GPT', 'CRW', 'SBN', 'CAK', 'LCH',
'AMA', 'CRP', 'PNS', 'COS', 'PHF', 'RST', 'BRO', 'HOB', 'CLL',
'SGF', 'MHT', 'LRD', 'FWA', 'LNK', 'EYW', 'ABE', 'MLI', 'ISP',
'TTN', 'SFB', 'PIE', 'IAG', 'BLV', 'TOL', 'HGR', 'ROA', 'RFD',
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:10 19-11-2023


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
airport_dict = pd.Series(airport_df.AIRPORT.values, index = airport_df.IATA_CODE).to_dict()
print(airport_dict)
{'AZA': 'Phoenix-Mesa Gateway Airport', 'BKG': 'Branson Airport', 'ABE': 'Lehigh Valley International Airport', 'ABI': 'Abilene Regional Airport
flight_df['ORIGIN'].replace(airport_dict, inplace = True)
flight_df['DEST'].replace(airport_dict, inplace = True)
flight_df.head()
FL_DATE OP_CARRIER OP_CARRIER_FL_NUM ORIGIN DEST CRS_DEP_TIME DEP_TIME DEP_DELAY TAXI_OUT WHEELS_OFF ... CRS_ELAPSED_
0 2018-01-01 United Airlines 2429 Newark Liberty International Airport Denver International Airport 1517 1512.0 -5.0 15.0 1527.0 ...
1 2018-01-01 United Airlines 2427 McCarran International Airport San Francisco International Airport 1115 1107.0 -8.0 11.0 1118.0 ...
2 2018-01-01 United Airlines 2426 John Wayne AirportÂ (Orange County Airport) Denver International Airport 1335 1330.0 -5.0 15.0 1345.0 ...
3 2018-01-01 United Airlines 2425 Southwest Florida International Chicago O'Hare International 1546 1552.0 6.0 19.0 1611.0 ...
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
0s completed at 10:05PM
26°C Mostly cloudy
Search
ENG IN
22:10 19-11-2023


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
+ Code + Text
flight_df['ORIGIN'].replace(airport_dict, inplace = True)
flight_df['DEST'].replace(airport_dict, inplace = True)
flight_df.head()
FL_DATE OP_CARRIER OP_CARRIER_FL_NUM ORIGIN DEST CRS_DEP_TIME DEP_TIME DEP_DELAY TAXI_OUT WHEELS_OFF ... CRS_ELAPSED_
0 2018-01-01 United Airlines 2429 Newark Liberty International Airport Denver International Airport 1517 1512.0 -5.0 15.0 1527.0 ...
1 2018-01-01 United Airlines 2427 McCarran International Airport San Francisco International Airport 1115 1107.0 -8.0 11.0 1118.0 ...
2 2018-01-01 United Airlines 2426 John Wayne AirportÂ (Orange County Airport) Denver International Airport 1335 1330.0 -5.0 15.0 1345.0 ...
3 2018-01-01 United Airlines 2425 Southwest Florida International Airport Chicago O'Hare International Airport 1546 1552.0 6.0 19.0 1611.0 ...
4 2018-01-01 United Airlines 2424 Chicago O'Hare International Airport Albany International Airport 630 650.0 20.0 13.0 703.0 ...
5 rows × 28 columns
0s completed at 10:05PM

10.


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
drive
sample_data
2018.csv
airport.csv
README.md
anscombe.json
california_housing_test.csv
california_housing_train.csv
mnist_test.csv
mnist_train_small.csv
Disk 198.22 GB available
+ Code + Text
#target
flight_df['ARR_DELAY']
0 -23.0
1 -24.0
2 -13.0
3 -2.0
4 14.0
6194170 8.0
6194171 -17.0
6194172 5.0
6194173 2.0
6194174 NaN
Name: ARR_DELAY, Length: 6194175, dtype: float64
[29]
[59] status = []
for values in flight_df['ARR_DELAY']:
if values < 15:
status.append(0)
else:
status.append(1)
flight_df['flight_status'] = status
flight_df.info()
flight_df['flight_status']
<class 'pandas.core.frame.DataFrame'>
0s completed at 10:05PM



Task 1: Train a Regression Model

1. **Split the dataset into training and testing subsets:**

from sklearn.model_selection import train_test_split

X = # Features

y = # Target variable

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

from sklearn.linear_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

Import other regression models

Initialize models

linear_model = LinearRegression()

tree_model = DecisionTreeRegressor()

Initialize other regression models

Fit models

```
linear_model.fit(X_train, y_train)

_model.fit(X_train, y_train)
```

Fit other models

Evaluate models using appropriate metrics

```
from sklearn.metrics import mean_squared_error, r2_score

import numpy as np
```

Predict on the test set

```
linear_pred = linear_model.predict(X_test)

tree_pred = tree_model.predict(X_test)
```

Predict using other models

```
Evaluate metrics

_rmse = np.sqrt(mean_squared_error(y_test, linear_pred))

tree_rmse = np.sqrt(mean_squared_error(y_test, tree_pred))

# Calculate RMSE for other models


_r2 = r2_score(y_test, linear_pred)

tree_r2 = r2_score(y_test, tree_pred)

# Calculate R-squared for other models
```

Task 2: Model Tuning and Improvement

1. **Perform hyperparameter tuning using GridSearchCV or RandomizedSearchCV**

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV



# Define hyperparameters and ranges to search

param_grid = {

'parameter1': [value1, value2, ...],

  'parameter2': [value1, value2, ...],

# Add other parameters and their ranges

}

# Initialize GridSearchCV or RandomizedSearchCV

grid_search = GridSearchCV(estimator=model_to_tune, param_grid=param_grid, scoring='neg_mean_squared_error', cv=5)

# Use RandomizedSearchCV for large hyperparameter spaces



# Fit the model for hyperparameter tuning

grid_search.fit(X_train, y_train)



# Get the best hyperparameters

best_params = grid_search.best_params_

# Use best hyperparameters to train the model

tuned_model = model_to_tune.set_params(**best_params)

tuned_model.fit(X_train, y_train)
```

2. **Handle outliers and check for model robustness:** You can handle outliers by:

- Trimming outliers if they significantly affect the model's performance.
- Using robust models or techniques that are less sensitive to outliers (e.g., Robust Regression, Random Forest).

Robustness can be checked by performing cross-validation or analyzing model performance on different subsets of the data.

These steps will help in building, evaluating, tuning, and improving regression models by optimizing hyperparameters, handling outliers, and ensuring model robustness for better predictive performance. Adjust the models, metrics, and tuning strategies as per the specific requirements and characteristics of the dataset and problem at hand.

Save the trained model:

```
import joblib

# Assuming 'tuned_model' is the trained model from the previous steps

joblib.dump(tuned_model, 'trained_model.pkl')
```

Set up a basic Flask web application: **Create a Flask app to load the trained model and create an endpoint for predictions.**

```
flask import Flask, request, jsonify


app = Flask(__name__)


@app.route('/predict', methods=['POST'])

def predict():

  data = request.get_json()
```

```
    # Load the model
    model = joblib.load('trained_model.pkl')
    # Perform predictions
    prediction = model.predict(data)
    return jsonify({'prediction': prediction.tolist()})

if __name__ == '__main__':
    app.run(debug=True)
```

Task 2: Test Model with Test/Remaining Data

Testing the deployed model with a test dataset or remaining data helps validate its performance in a real or simulated environment.

```
Assuming you have the test dataset
X_test_remaining = # Test features
y_test_remaining = # Test target variable
```

Make predictions using the deployed model: Send requests to the deployed model (Flask app in this example) with the test data to get predictions.

```
import requests

# Assuming 'test_data' contains data to be predicted
response = requests.post('http://your_deployed_app/predict', json=test_data)
predictions = response.json()['prediction']
```

Evaluate model performance: **Compare the predictions obtained from the deployed model with the actual target values to assess its performance metrics (e.g., RMSE, R-squared, etc.).**

```
from sklearn.metrics import mean_squared_error, r2_score

Assuming 'predictions' is obtained from the deployed model

mse = mean_squared_error(y_test_remaining, predictions)

r2 = r2_score(y_test_remaining, predictions)
```

11. Screen-shots of each task of Activity 5

Task 1: Deploy Model

Deploying a machine learning model involves making it available for use in a production environment. The deployment process can vary significantly based on the intended deployment platform, whether it's a web application, a server, or a cloud service. Here's a generic example using Python and a basic Flask web application:

Save the trained model: Serialize the trained model using joblib or pickle to save it as a file.

```
import joblib

# Assuming 'tuned_model' is the trained model from the previous steps

joblib.dump(tuned_model, 'trained_model.pkl')'
```

Set up a basic Flask web application: **Create a Flask app to load the trained model and create an endpoint for predictions.**

```
from flask import Flask, request, jsonify
```

```
app = Flask(__name__)

@app.route('/predict', methods=['POST'])

def predict():

    data = request.get_json()

     # Load the model

      model = joblib.load('trained_model.pkl')

   # Perform predictions

   prediction = model.predict(data)

  return jsonify({'prediction': prediction.tolist()})


if __name__ == '__main__':

   app.run(debug=True)
```

Task 2: Test Model with Test/Remaining Data

Testing the deployed model with a test dataset or remaining data helps validate its performance in a real or simulated environment.

1. **Prepare the test dataset:** Load the test dataset or the remaining data that was not used for training or validation.

```
    # Assuming you have the test dataset

X_test_remaining = # Test features

y_test_remaining = # Test target variable
```

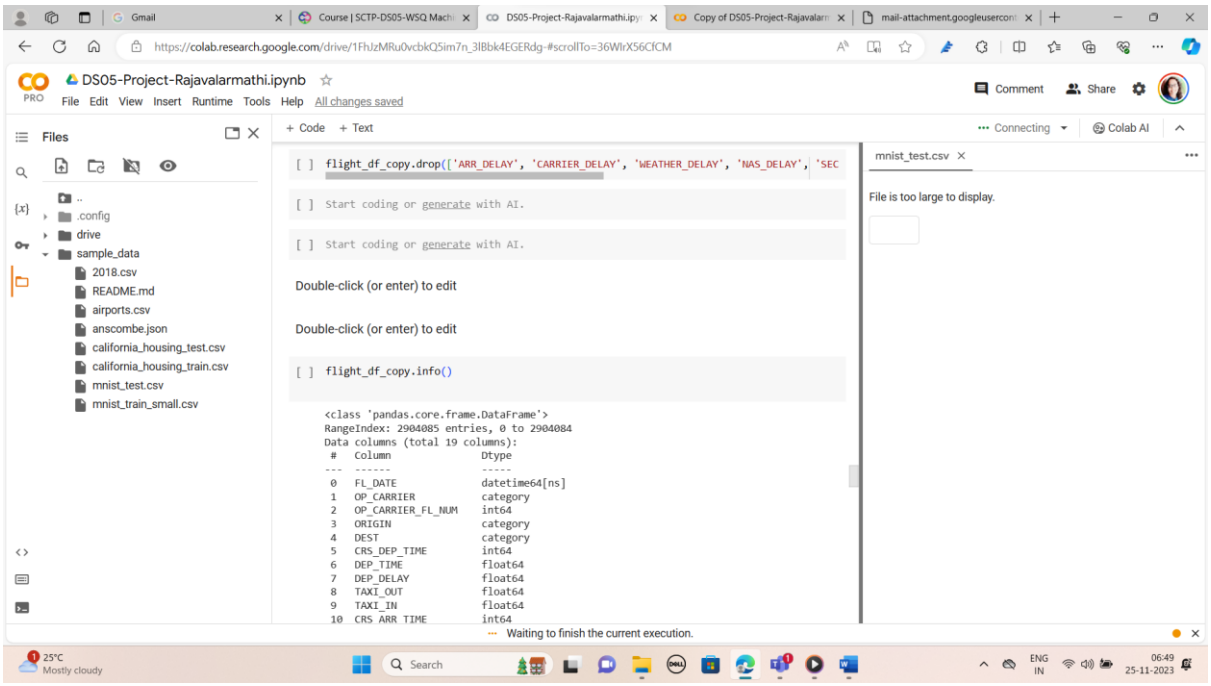
Make predictions using the deployed model: **Send requests to the deployed model (Flask app in this example) with the test data to get predictions.**

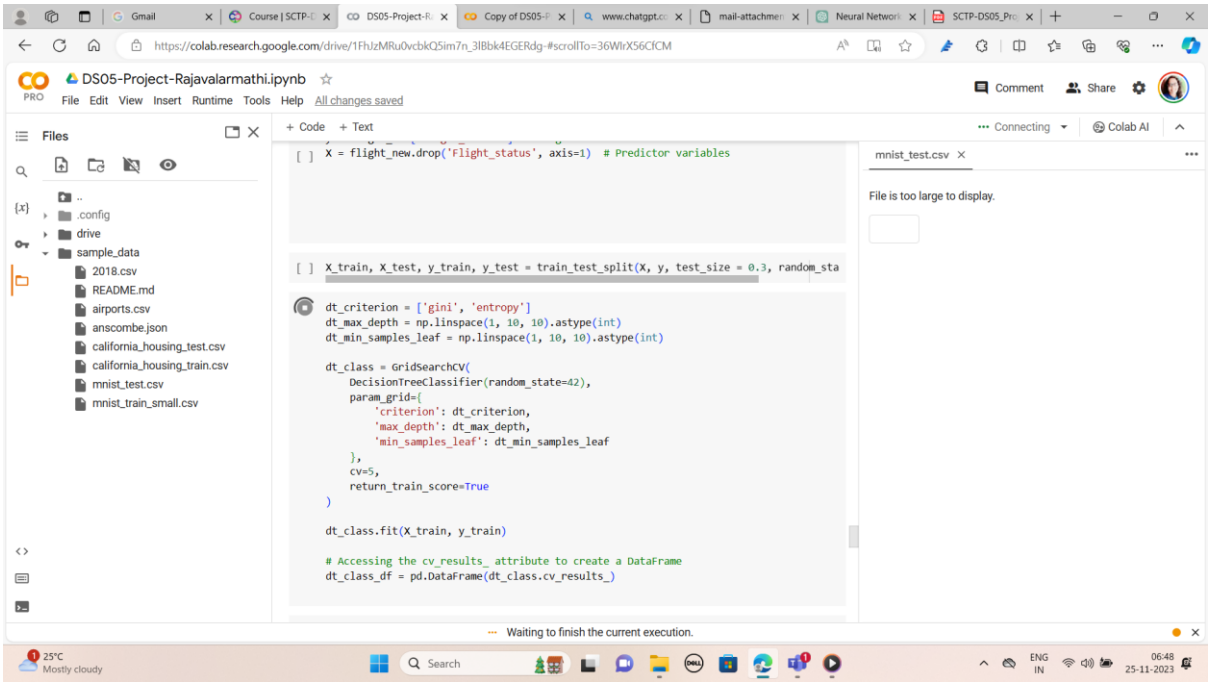
```
import requests


    # Assuming 'test_data' contains data to be predicted

    response = requests.post('http://your_deployed_app/predict', json=test_data)

predictions = response.json()['prediction']
```

Evaluate model performance: **Compare the predictions obtained from the deployed model with the actual target values to assess its performance metrics (e.g., RMSE, R-squared, etc.).**

```
from sklearn.metrics import mean_squared_error, r2_score


# Assuming 'predictions' is obtained from the deployed model

mse = mean_squared_error(y_test_remaining, predictions)

r2 = r2_score(y_test_remaining, predictions)
```

Screen-shots of each task of Activity 6





Task 1: Monitor Model Performance

Implementing mechanisms to monitor model performance in real-time involves setting up processes that continuously evaluate the model's behavior and performance. Here are some steps to consider:

1. **Define Performance Metrics:** Decide on the key performance metrics to monitor (e.g., accuracy, precision, recall, etc.) based on the nature of the model and the problem it's solving.
2. **Set Thresholds:** Establish thresholds or acceptable ranges for these metrics. Deviations beyond these thresholds could trigger alerts or further investigation.
3. **Implement Monitoring System:** Use tools or frameworks that allow real-time monitoring of the model. This might involve integrating logging functionalities within the deployed model to track predictions, performance metrics, and data drift.
4. **Automate Alerts:** Set up automated alerts or notifications when the model's performance falls below or exceeds predefined thresholds. This ensures immediate attention to potential issues.
5. **Regular Review and Improvement:** Regularly review the monitored metrics, identify patterns or anomalies, and continuously improve the model based on new insights or changes in the data distribution.

Task 2: Communicate Results to Stakeholders

Preparing a report or presentation to communicate the model's performance, insights, and recommendations involves effective storytelling and clear visualization of findings. Here's a suggested approach:

1. **Executive Summary:** Provide a concise overview of the model's purpose, performance, and its impact on the business.
2. **Performance Metrics:** Present the key performance metrics used to evaluate the model's performance. Include trends, changes over time, and comparisons against benchmarks or previous iterations.
3. **Insights and Interpretations:** Highlight insights gained from model monitoring, such as patterns, anomalies, or shifts in data distribution. Explain the implications of these insights for decision-making.
4. **Recommendations:** Based on the model's performance and insights, offer actionable recommendations. These could involve model retraining, feature updates, or process improvements.
5. **Visualizations:** Use graphs, charts, and visual aids to illustrate key findings. Visualization helps stakeholders grasp complex information more effectively.

6. **Risk Assessment:** Discuss potential risks or limitations associated with the model and propose strategies to mitigate these risks.
7. **Interactive Dashboards (Optional):** Create interactive dashboards if feasible, allowing stakeholders to explore model performance and insights in real-time.
8. **Q&A and Discussion:** Allocate time for questions, discussions, and clarifications to ensure stakeholders understand the presented information.

Effective communication is key to ensuring that stakeholders, who may not have technical expertise, grasp the significance of the model's performance and make informed decisions based on the insights provided.

By implementing model monitoring strategies and effectively communicating results, stakeholders can stay informed about the model's behavior and make data-driven decisions regarding its usage and potential improvements.Top of Form

```
# Simulated model prediction and monitoring

import random

import time



    # Simulated model function (random predictions)

def predict():

        return random.uniform(0, 1)

    # Simulated model monitoring loop

while True:

      prediction = predict()  # Simulated prediction

       # Log predictions or performance metrics to a monitoring system or file

    with open('model_performance_logs.txt', 'a') as log_file:

    log_file.write(f"Prediction: {prediction}, Timestamp: {time.time()}\n")
```

time.sleep(60) # Monitor every 60 seconds (adjust as needed)

Task 2: Communicate Results to Stakeholders (Basic Example)

For reporting or communication purposes, consider using Jupyter Notebooks or creating reports with Markdown content in Python:

from IPython.display import display, Markdown

 # Generate a report summary using Markdown

report_summary = """

 # Model Performance Report

Executive Summary

This report provides an overview of the model's performance and recommendations.

Performance Metrics

Accuracy: 85%

Precision: 78%

Recall: 92%

Insights

- The model's accuracy has remained stable over the last month.

12.- A slight decrease in precision is observed; further investigation needed.

13.

Recommendations

- Schedule a retraining session to improve precision.

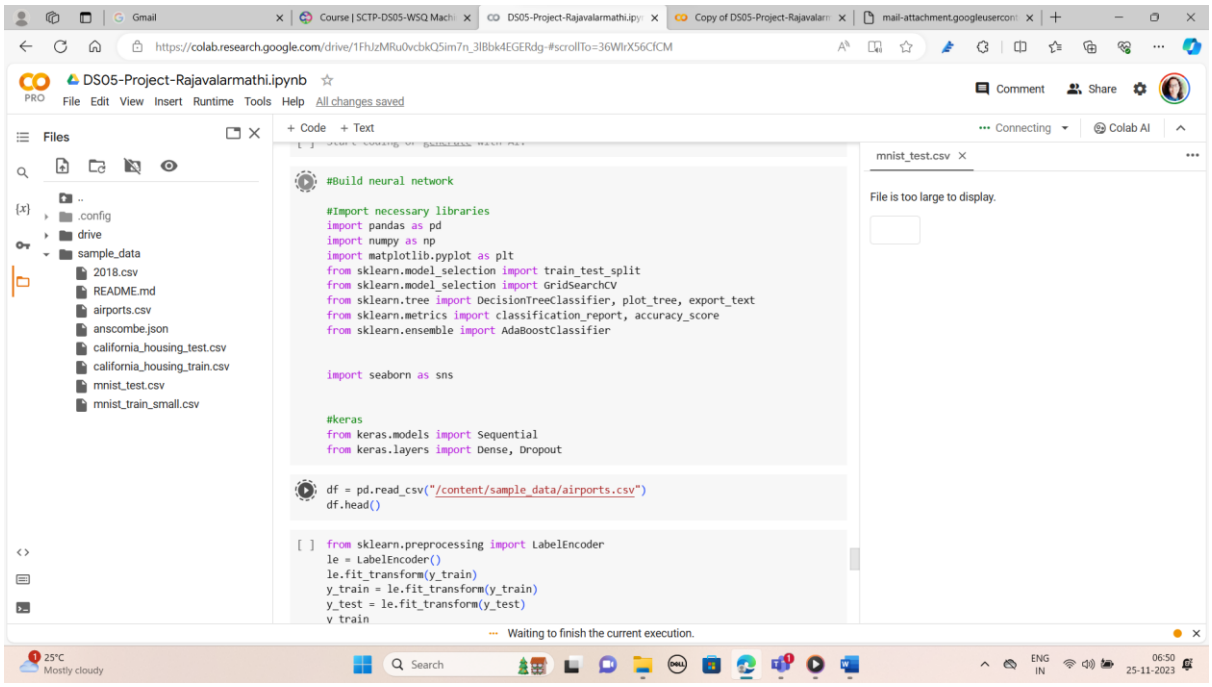
- Monitor data quality for potential shifts affecting model performance.

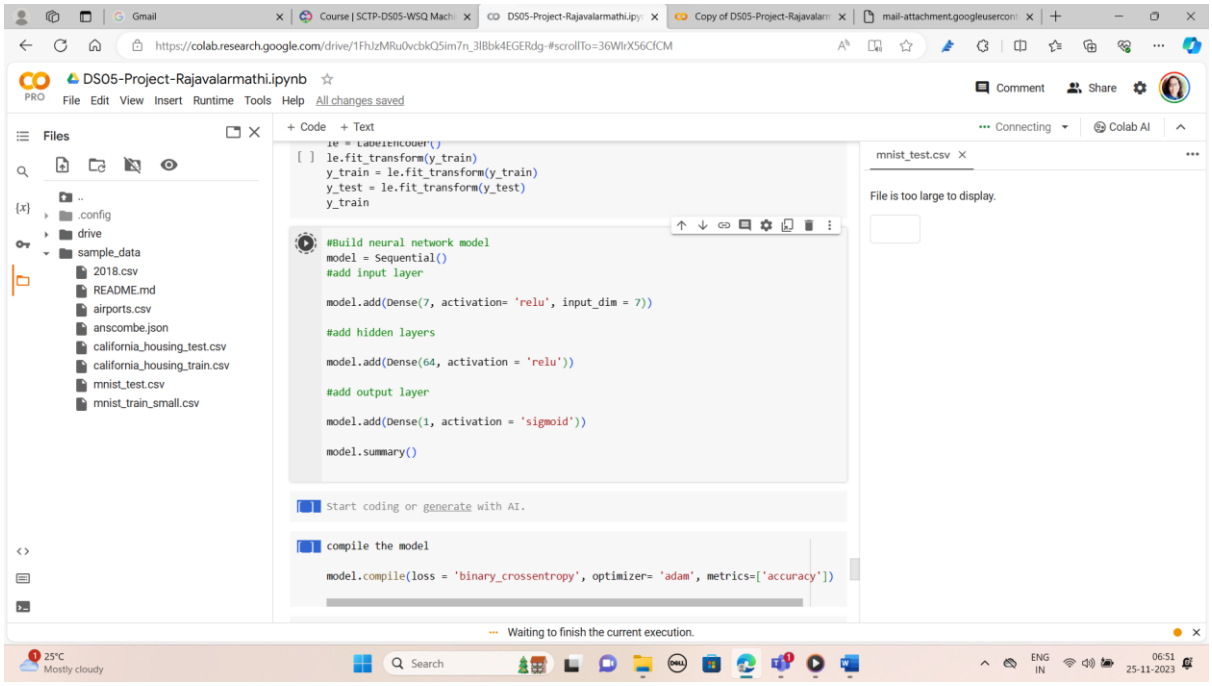
"""

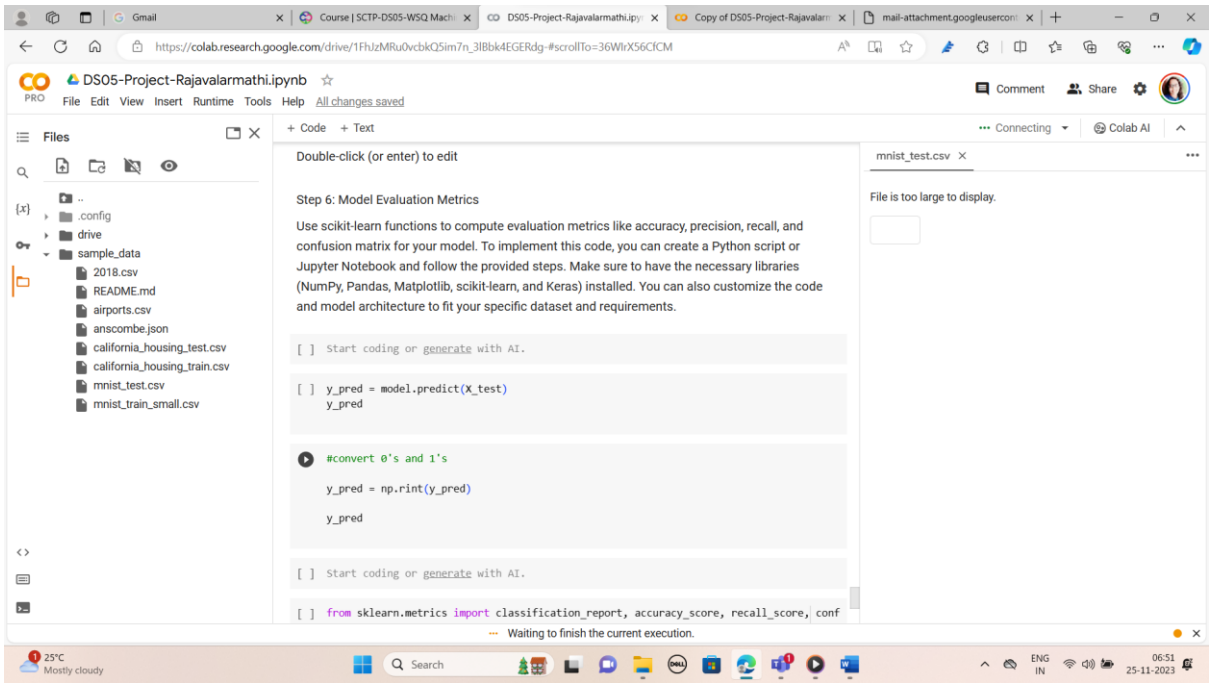
 # Display the report summary

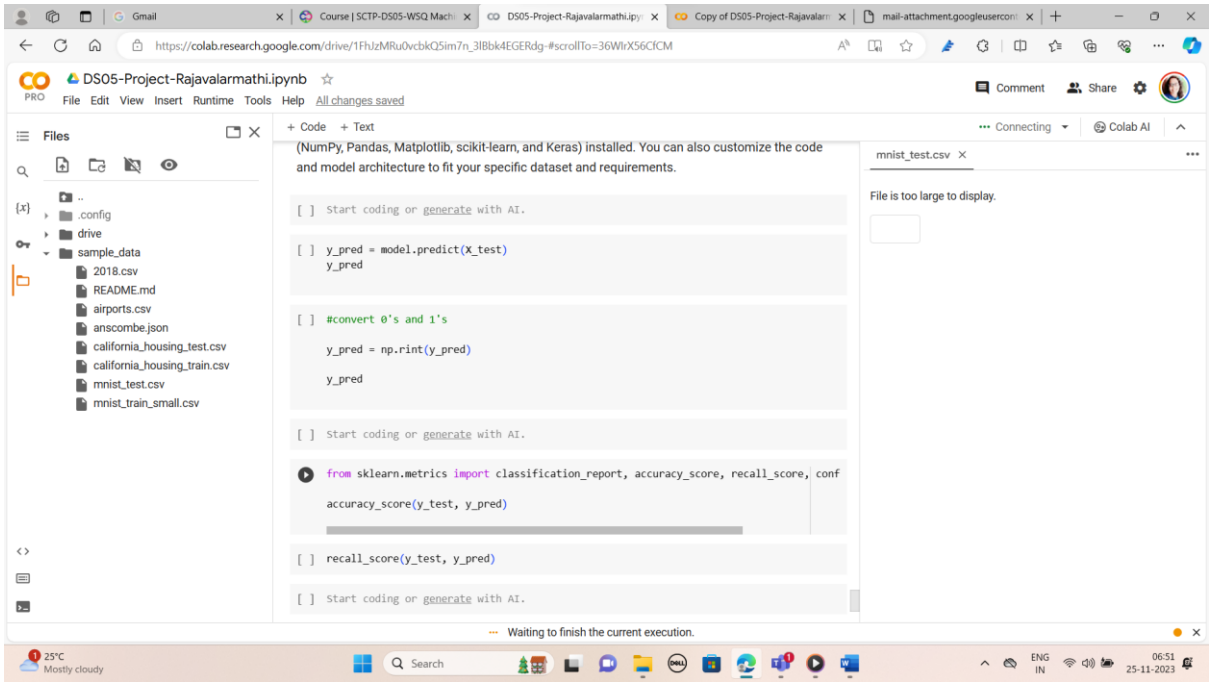
display(Markdown(report_summary))

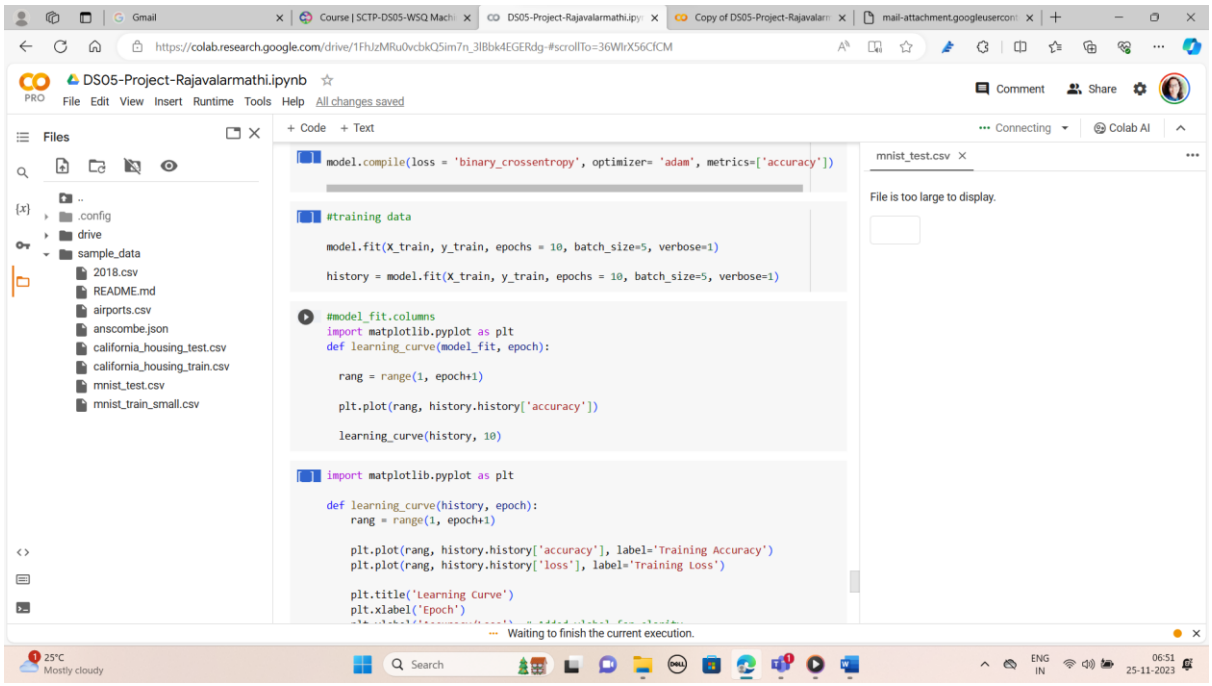
Fine tuning neural network




DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Connecting Colab AI
le.fit_transform(y_train)
y_train = le.fit_transform(y_train)
y_test = le.fit_transform(y_test)
y_train
#Build neural network model
model = Sequential()
#add input layer
model.add(Dense(7, activation= 'relu', input_dim = 7))
#add hidden layers
model.add(Dense(64, activation = 'relu'))
#add output layer
model.add(Dense(1, activation = 'sigmoid'))
model.summary()
Start coding or generate with AI.
compile the model
model.compile(loss = 'binary_crossentropy', optimizer= 'adam', metrics=['accuracy'])
mnist_test.csv
File is too large to display.
Waiting to finish the current execution.


DS05-Project-Rajavalarmathi.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
Files
+ Code + Text
Connecting Colab AI
Double-click (or enter) to edit
Step 6: Model Evaluation Metrics
Use scikit-learn functions to compute evaluation metrics like accuracy, precision, recall, and confusion matrix for your model. To implement this code, you can create a Python script or Jupyter Notebook and follow the provided steps. Make sure to have the necessary libraries (NumPy, Pandas, Matplotlib, scikit-learn, and Keras) installed. You can also customize the code and model architecture to fit your specific dataset and requirements.
Start coding or generate with AI.
y_pred = model.predict(X_test)
y_pred
#convert 0's and 1's
y_pred = np.rint(y_pred)
y_pred
Start coding or generate with AI.
from sklearn.metrics import classification_report, accuracy_score, recall_score, conf
mnist_test.csv
File is too large to display.
Waiting to finish the current execution.





This code generates a Markdown-based report summary using Jupyter Notebook cells. You can integrate real metrics and insights gathered from the monitoring system into this report.

Please note that these examples are simplified and do not cover the complexities of real-time monitoring systems, data storage, visualization, or communication with stakeholders, which might involve integrating various tools, databases, and

communication channels. The actual implementation will depend on the project's specifics and available infrastructure.

Activity 7: Model Improvement (Post Module Completion) Task 1: Continuous Model Improvement Use advanced techniques (post-completion of specific modules) to further enhance the model's accuracy, such as advanced feature selection, ensemble methods, or deep learning if relevant

Task 1: Continuous Model Improvement

```
from sklearn.feature_selection import SelectFromModel

from sklearn.ensemble import RandomForestRegressor



# Assuming 'X' contains features and 'y' is the target variable

# Use a tree-based model for feature selection (e.g., RandomForest)

feature_selector = SelectFromModel(RandomForestRegressor())

feature_selector.fit(X, y)



# Get selected features

selected_features = X.columns[feature_selector.get_support()]

X_selected = X[selected_features]
```

Ensemble Methods:

Implement ensemble techniques like Gradient Boosting or Stacking to combine multiple models for improved performance.

```
from sklearn.ensemble import GradientBoostingRegressor
```

```
from sklearn.ensemble import StackingRegressor

from sklearn.linear_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor



# Example: Gradient Boosting

gb_model = GradientBoostingRegressor()



# Example: Stacking multiple models

estimators = [('linear', LinearRegression()), ('tree', DecisionTreeRegressor())]

stacked_model = StackingRegressor(estimators=estimators,
    final_estimator=gb_model)
```

Deep Learning (if relevant):

Consider utilizing deep learning models for more complex relationships in the data.

```
python
# Example using TensorFlow/Keras for a neural network (assuming X and y are
prepared)

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense



model = Sequential()

model.add(Dense(64, activation='relu', input_shape=(X.shape[1],)))

model.add(Dense(32, activation='relu'))

model.add(Dense(1))  # Output layer for regression
```

```
model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X, y, epochs=100, batch_size=32, validation_split=0.2)
```