Applied Machine learning
Lab File

A

Lab File

Submitted for Applied Machine Learning

Lab Evaluation

By

Vidushi Gupta

Sap ID: *500123459*

Batch: 13

To

Dr. Abhijit Kumar

UPES

UNIVERSITY OF TOMORROW

SCHOOL OF COMPUTER SCIENCE

AIML CLUSTER

UNIVERSITY OF PETROLEUM AND ENERGY STUDIES

Index

Experiment No.	Experiment Name	Experiment Date	Submission Date	Signature
1	Setting the Environment			
2-3	Data Preprocessing & Application of Data Preprocessing ***Datasets:*** • Predicting Housing prices • Employee Dataset • Government Dataset			
4-5	Model Development & Hyper Parameter Tuning for Regression Problems ***Datasets:*** • Predicting Housing Prices • Employee Dataset • Government Dataset			
6-7	Classification Problem ***Datasets:*** • Iris Flower Classification • Breast Cancer Diagnosis • Handwritten Digit Recognition – MNIST • Bank Churn			
8-9	Clustering Problems ***Datasets:*** • Disease Diagnosis from Medical Images • DBSCAN			

Contents

Experiment – 1 1

Setting The Environment 1

Step 1: Logging into Google Collab 1

Step 2: Creating a New Notebook 1

Step 3: Changing the Runtime Type 2

Step 4: Installing and Importing Python Libraries and Packages 2

Libraries In Python: 3

Experiment: 2 -3 6

Data Preprocessing 6

- *Housing Dataset* 6
- *Employee Dataset* 9
- *Government Dataset* 12

Experiment: 4-5 14

Model Development & Hyper Parameter Tuning for Regression Problems 14

- *Housing Dataset* 14
- *Employee Dataset* 16
- *Government Dataset* 17

Experiment: 6-7 19

Classification Problem 19

- *Iris Flower Classification* 19
- *Breast Cancer Diagnosis* 20
- *Handwritten Digit Recognition – MNIST* 21
- *Bank Churn Dataset* 22

Experiment: 8-9 23

- *Disease Diagnosis from Medical Images* 23
- *DBSCAN* 24

Experiment – 1
Setting The Environment

Google Colab is a cloud-based Jupyter notebook environment that allows users to write and execute Python code in the cloud without installing anything on their local machine. It is widely used for machine learning (ML), data science, and deep learning projects.

The process of setting up the environment in Google Colab, including logging in, creating a new notebook, changing the runtime type, setting up a virtual environment, installing Python libraries, and saving the environment to Google Drive.

Step 1: Logging into Google Collab

- **Steps to Log in to Google Collab:**

1. Open your web browser.
2. Go to **Google Collaboratory**.
3. Click **"Sign in"** in the top right corner if you are not already logged into your Google account.
4. Once logged in, you will see the **Google Colab welcome page** with several options for opening a notebook.

Step 2: Creating a New Notebook

A **notebook** in Google Colab is, where you can write and execute Python code in separate cells.

2.1 Steps to Create a New Notebook:

1. After logging into Google Colab, click on **"File"** then **"New Notebook."**
2. A new notebook will open, containing:
    - A **code cell** for Python code and to execute it.
    - A **text cell** to write Markdown-formatted documentation.

3. Rename the notebook by clicking on **"Untitled.ipynb"** at the top and entering a new name

Step 3: Changing the Runtime Type

By default, Google Colab runs on a CPU. If your project requires more computational power, you can **enable GPU or TPU acceleration**.

Click **"Save."**

Step 4: Installing and Importing Python Libraries and Packages

Google Colab comes pre-installed with many Python libraries, but some specialized packages may need to be installed manually.

4.1 Installing Libraries

- Install a package using pip:

Example : pip install scikit-learn

- To install multiple packages at once:

4.2 Importing Libraries

After installing, you can **import the libraries** in Python:

Example :

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

4.3 Checking Installed Libraries

Check if a package is installed

Step 5: Saving Your Environment to Google Drive

Google Colab allows you to **save your work and datasets** in Google Drive for future use.

5.1 Mount Google Drive

To access Google Drive from Colab, run:

```
from google.colab import drive
drive.mount('/content/drive')
```

- After running the command, a **link** will appear.
- Click the link, **authorize Google Drive access**, and copy the authentication code back into Colab.
- Your Google Drive is now accessible at:

5.2 Saving Your Notebook to Google Drive

- Click **"File"** then **"Save a copy in Drive."**
- A copy will be stored in **Google Drive** then **Colab Notebooks**.

5.3 Loading a Dataset from Google Drive

```
import pandas as pd
df = pd.read_csv('/content/drive/My Drive/dataset.csv')
```

Libraries In Python:

- **NumPy (Numerical Python)**
  It is a fundamental library for numerical computing in Python, providing support for multi-dimensional arrays and matrices, along with a collection of mathematical functions to perform efficient computations. It is widely used in scientific computing, data analysis, and machine learning due to its ability to handle large datasets and perform vectorized operations efficiently. NumPy also supports functions for linear algebra, Fourier transforms, and random number generation, making it essential for high-performance computing.

- **Pandas**
  It is a powerful data manipulation and analysis library built on top of NumPy. It provides two main data structures: Series (1D) and DataFrame (2D), which allow users to efficiently organize, clean, transform, and analyze data. Pandas supports operations such as reading and writing files (CSV, Excel, SQL, JSON),

handling missing values, filtering data, and performing group-based operations. It is widely used in data preprocessing for machine learning models and business analytics, as it simplifies working with structured data.

- **Matplotlib**
  It is a widely used data visualization library that enables the creation of line plots, bar charts, scatter plots, histograms, and more. It provides extensive customization options for labels, titles, colors, legends, and grid lines, making it an essential tool for exploratory data analysis (EDA). Matplotlib allows users to visualize trends, distributions, and relationships between variables, helping to gain insights from data before applying machine learning models.

- **Scikit-learn**
  It is a comprehensive machine learning library built on NumPy and Pandas. It offers a variety of tools for classification, regression, clustering, dimensionality reduction, and model evaluation. Additionally, it includes functionalities for data preprocessing, feature selection, hyperparameter tuning, and cross-validation, making it an essential tool for building and optimizing machine learning models. Scikit-learn is widely used for tasks such as spam detection, sentiment analysis, fraud detection, and recommendation systems.

- **SciPy (Scientific Python)**
  It extends NumPy's capabilities by providing additional tools for scientific and technical computing. It includes modules for optimization, integration, signal processing, image processing, and statistical analysis. SciPy is particularly useful in scientific research, engineering, and medical image processing, where complex mathematical computations are required. It is also widely used for hypothesis testing, probability distributions, and solving differential equations, making it a valuable tool for advanced analytics and computational applications.

- **Seaborn**

  Seaborn is a high-level data visualization library built on top of Matplotlib. It provides an easy way to create attractive and informative statistical graphics, making it particularly useful for exploratory data analysis (EDA). Seaborn simplifies the process of creating complex plots like heatmaps, violin plots, box plots, and pair plots, which help understand data distributions and correlations. Unlike Matplotlib, Seaborn has built-in support for data frames, allowing users to directly visualize Pandas datasets without additional data manipulation.

Experiment: 2 -3

Data Preprocessing

- ***Housing Dataset***

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
#Ishita garg 500122821
```

```python
url = "https://raw.githubusercontent.com/ageron/handson-ml2/master/datasets/housing/housing.csv"
df = pd.read_csv(url)
#Ishita garg 500122821
```

```python
print("Dataset Preview:")
display(df.head())
print("\nDataset Information:")
df.info()
#Ishita garg 500122821
```

Dataset Preview:

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value	ocean_proximity
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0	NEAR BAY
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0	NEAR BAY
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0	NEAR BAY
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0	NEAR BAY
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0	NEAR BAY

```
Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
memory usage: 1.6+ MB
```

```python
df['rooms_per_household'] = df['total_rooms'] / df['households']
df['bedrooms_ratio'] = df['total_bedrooms'] / df['total_rooms']
df['population_per_household'] = df['population'] / df['households']
df['income_per_capita'] = df['median_income'] / (df['population'] / df['households'])

df['bedrooms_per_household'] = df['total_bedrooms'] / df['households']
df['population_per_room'] = df['population'] / df['total_rooms']
df['income_per_household'] = df['median_income'] * df['households'] / df['population']
df['rooms_per_person'] = df['total_rooms'] / df['population']
df['bedrooms_per_person'] = df['total_bedrooms'] / df['population']
```

```python
print("\nMissing values before handling:")
print(df.isnull().sum())
#Ishita garg 500122821
```

```
Missing values before handling:
longitude                     0
latitude                      0
housing_median_age            0
total_rooms                   0
total_bedrooms              207
population                    0
households                    0
median_income                 0
median_house_value            0
ocean_proximity               0
rooms_per_household           0
bedrooms_ratio              207
population_per_household      0
income_per_capita             0
bedrooms_per_household      207
population_per_room           0
income_per_household          0
rooms_per_person              0
bedrooms_per_person         207
dtype: int64
```

```
numeric_columns = df.select_dtypes(include=['number']).columns
df[numeric_columns] = df[numeric_columns].fillna(df[numeric_columns].media

print("\nMissing values after handling:")
print(df.isnull().sum())

#Ishita garg 500122821
```

```
Missing values after handling:
longitude                   0
latitude                    0
housing_median_age          0
total_rooms                 0
total_bedrooms              0
population                  0
households                  0
median_income               0
median_house_value          0
ocean_proximity             0
rooms_per_household         0
bedrooms_ratio              0
population_per_household    0
income_per_capita           0
bedrooms_per_household      0
population_per_room         0
income_per_household        0
rooms_per_person            0
bedrooms_per_person         0
dtype: int64
```

```
df_processed = pd.get_dummies(df, columns=['ocean_proximity'], drop_first=True)

# Feature Selection
X = df_processed.drop("median_house_value", axis=1)
y = df_processed["median_house_value"]

#Ishita garg 500122821
```

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

#Ishita garg 500122821
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

#Ishita garg 500122821
```

```
model = LinearRegression()
model.fit(X_train_scaled, y_train)

#Ishita garg 500122821
```

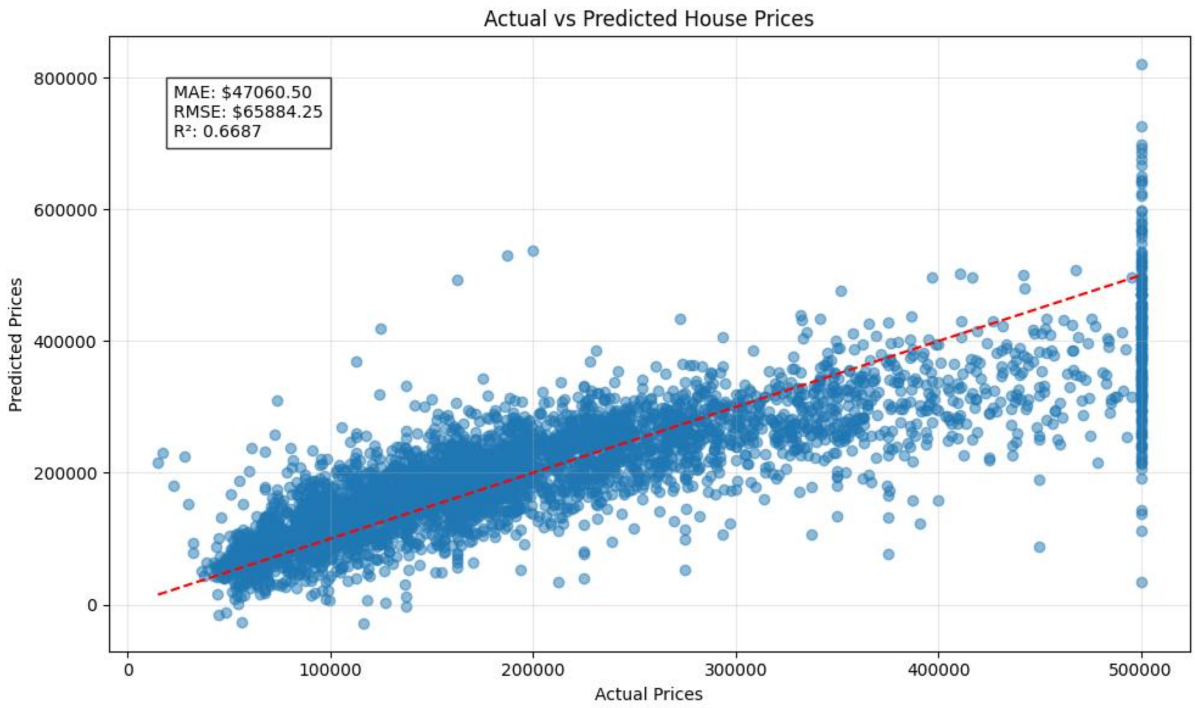
```
LinearRegression
LinearRegression()
```

```
y_pred = model.predict(X_test_scaled)

#Ishita garg 500122821
```

```
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("\nModel Evaluation Metrics:")
print(f"Mean Absolute Error: {mae:.2f}")
print(f"Mean Squared Error: {mse:.2f}")
print(f"Root Mean Squared Error: {rmse:.2f}")
print(f"R² Score: {r2:.4f}")

#Ishita garg 500122821
```

```
Model Evaluation Metrics:
Mean Absolute Error: 47060.50
Mean Squared Error: 4340734574.69
Root Mean Squared Error: 65884.25
R² Score: 0.6687
```

```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted House Prices")
plt.grid(True, alpha=0.3)

# Add text for evaluation metrics
plt.figtext(0.15, 0.8, f"MAE: ${mae:.2f}\nRMSE: ${rmse:.2f}\nR²: {r2:.4f}",
            bbox=dict(facecolor='white', alpha=0.8))

plt.tight_layout()
plt.show()

#Ishita garg 500122821
```



- ***<u>Employee Dataset</u>***

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    from sklearn.model_selection import train_test_split
    from sklearn.preprocessing import StandardScaler
    from sklearn.ensemble import RandomForestRegressor
    from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
    #Ishita Garg 500122821
```

```
[ ] url = "https://raw.githubusercontent.com/mwaskom/seaborn-data/master/exercise.csv"
    df = pd.read_csv(url)

    #Ishita Garg 500122821
```

```
print("Dataset shape:", df.shape)
print("\nColumns:", df.columns.tolist())
print("\nFirst few rows:")
print(df.head())
print("\nData types:")
print(df.dtypes)
print("\nMissing values:")
print(df.isnull().sum())

#Ishita Garg 500122821
```

```
Dataset shape: (90, 6)

Columns: ['Unnamed: 0', 'id', 'diet', 'pulse', 'time', 'kind']

First few rows:
   Unnamed: 0  id     diet  pulse    time  kind
0           0   1  low fat     85   1 min  rest
1           1   1  low fat     85  15 min  rest
2           2   1  low fat     88  30 min  rest
3           3   2  low fat     90   1 min  rest
4           4   2  low fat     92  15 min  rest

Data types:
Unnamed: 0     int64
id             int64
diet          object
pulse          int64
time          object
kind          object
dtype: object
```

```
Missing values:
Unnamed: 0    0
id            0
diet          0
pulse         0
time          0
kind          0
dtype: int64
```

```python
for column in df.columns:
    if df[column].dtype == 'object':
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        df[column].fillna(df[column].median(), inplace=True)

#Ishita Garg 500122821
```

```
<ipython-input-5-e7e0c6e0e099>:5: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df[column].fillna(df[column].median(), inplace=True)
<ipython-input-5-e7e0c6e0e099>:3: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


  df[column].fillna(df[column].mode()[0], inplace=True)
```

```python
if 'monthly_income' in df.columns:
    target_column = 'monthly_income'
    feature_columns = [col for col in df.columns if col != target_column]
    df_selected = df[feature_columns + [target_column]]
else:
    target_column = 'pulse'
    feature_columns = [col for col in df.columns if col != target_column]
    df_selected = df[feature_columns + [target_column]]

#Ishita Garg 500122821
```

```python
categorical_columns = df_selected.select_dtypes(include=['object']).columns
if len(categorical_columns) > 0:
    df_encoded = pd.get_dummies(df_selected, columns=categorical_columns, drop_first=True)
else:
    df_encoded = df_selected.copy()

#Ishita Garg 500122821
```

```python
X = df_encoded.drop(target_column, axis=1)
y = df_encoded[target_column]

#Ishita Garg 500122821
```

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

#Ishita Garg 500122821
```

```python
scaler = StandardScaler()
numerical_cols = X_train.select_dtypes(include=['float64', 'int64']).columns
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()
X_train_scaled[numerical_cols] = scaler.fit_transform(X_train[numerical_cols])
X_test_scaled[numerical_cols] = scaler.transform(X_test[numerical_cols])

#Ishita Garg 500122821
```

```python
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)

#Ishita Garg 500122821
```

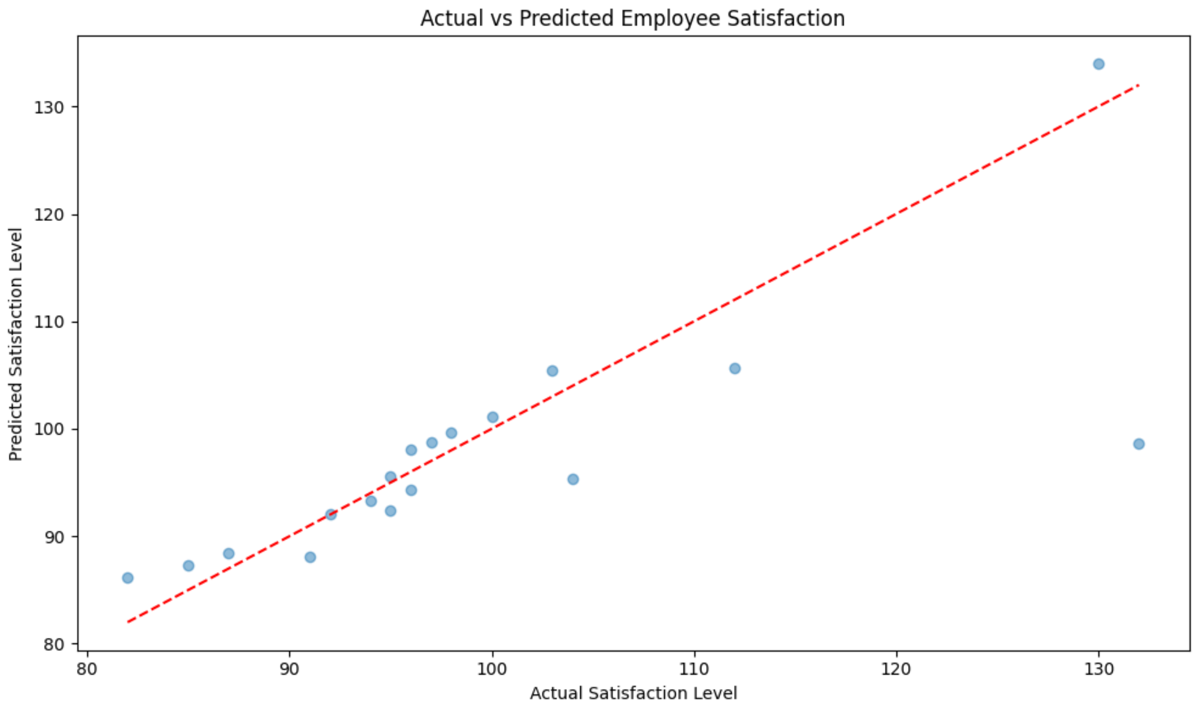
```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```python
y_pred = model.predict(X_test_scaled)

#Ishita Garg 500122821
```

```python
print("\nModel Evaluation:")
print(f"MAE: {mean_absolute_error(y_test, y_pred)}")
print(f"MSE: {mean_squared_error(y_test, y_pred)}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred))}")
print(f"R2 Score: {r2_score(y_test, y_pred)}")

#Ishita Garg 500122821
```

```
Model Evaluation:
MAE: 4.327222222222222
MSE: 72.39056111111113
RMSE: 8.508264283102113
R2 Score: 0.5769764306970872
```

```python
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'r--')
plt.xlabel("Actual Satisfaction Level")
plt.ylabel("Predicted Satisfaction Level")
plt.title("Actual vs Predicted Employee Satisfaction")
plt.tight_layout()
plt.show()

#Ishita Garg 500122821
```



- ***<u>Government Dataset</u>***

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.preprocessing import RobustScaler
from sklearn.pipeline import Pipeline
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import warnings
warnings.filterwarnings('ignore')

#Ishita Garg 500122821
```

```
[2] df = pd.read_csv('/content/Cities1.csv')
    df.drop(columns=['S. No.'], errors='ignore', inplace=True)

    #Ishita Garg 500122821
```

```
[3] df.fillna(df.median(numeric_only=True), inplace=True)
    for col in df.select_dtypes(include=['object']):
        df[col].fillna(df[col].mode()[0], inplace=True)

    #Ishita Garg 500122821
```

```
[4] df = pd.get_dummies(df, drop_first=True)

    #Ishita Garg 500122821
```

```
[5] X = df.drop(["AirQuality"], axis=1)
    y = df["AirQuality"]

    #Ishita Garg 500122821
```

```
[6] rf_params = {'n_estimators': [50, 100], 'max_depth': [None, 10]}
    rf = GridSearchCV(RandomForestRegressor(random_state=42, n_jobs=-1), rf_params, cv=2, n_jobs=-1)

    #Ishita Garg 500122821
```

```
[7] kf = KFold(n_splits=2, shuffle=True, random_state=42)
    scores, preds, actuals = [], [], []

    #Ishita Garg 500122821
```

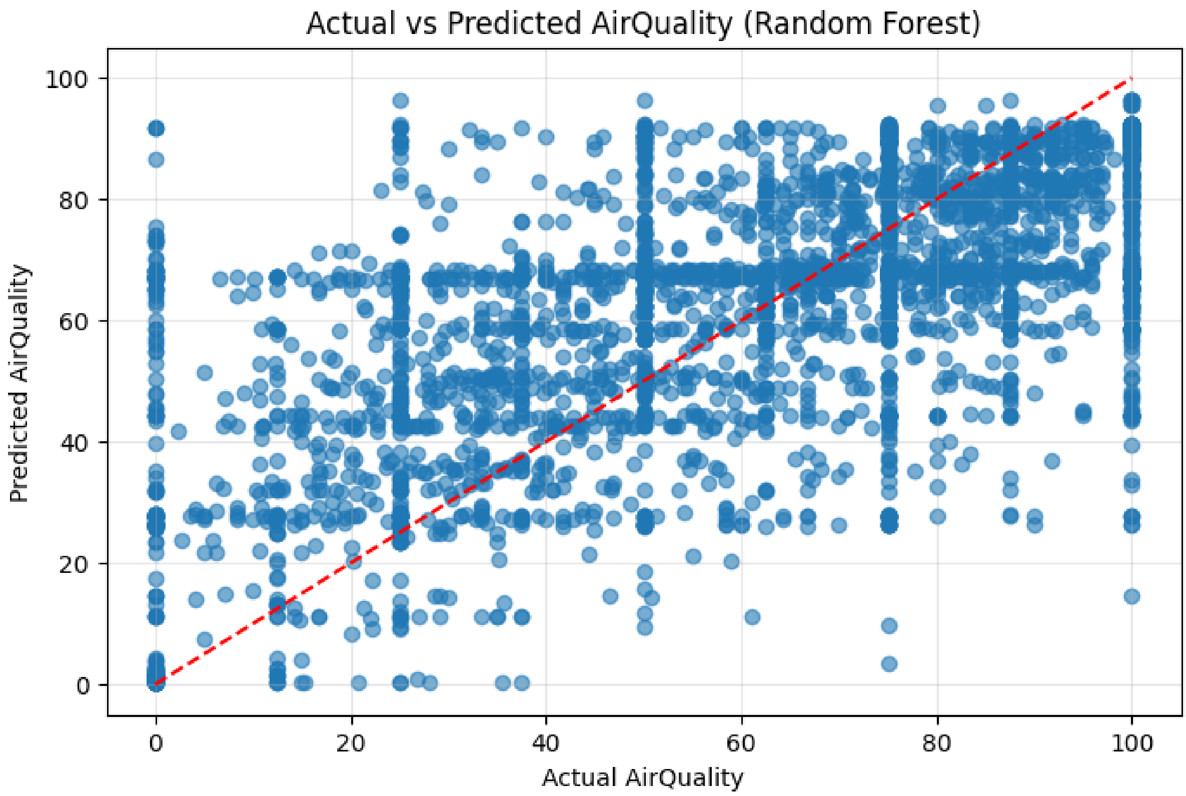
```
print("\nTraining Random Forest for AirQuality:")

pipeline = Pipeline([("scaler", RobustScaler()), ("rf", rf)])

for train_idx, test_idx in kf.split(X):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)

    scores.append(r2_score(y_test, y_pred))
    preds.extend(y_pred)
    actuals.extend(y_test)

#Ishita Garg 500122821
```

```
Training Random Forest for AirQuality:
```

```
avg_r2 = np.mean(scores)
print(f"  Avg R²: {avg_r2:.4f}")

#Ishita Garg 500122821
```

```
  Avg R²: 0.4754
```

```
plt.figure(figsize=(8, 5))
plt.scatter(actuals, preds, alpha=0.6)
plt.plot([min(actuals), max(actuals)], [min(actuals), max(actuals)], 'r--')
plt.xlabel("Actual AirQuality")
plt.ylabel("Predicted AirQuality")
plt.title("Actual vs Predicted AirQuality (Random Forest)")
plt.grid(alpha=0.3)
plt.show()

#Ishita Garg 500122821
```

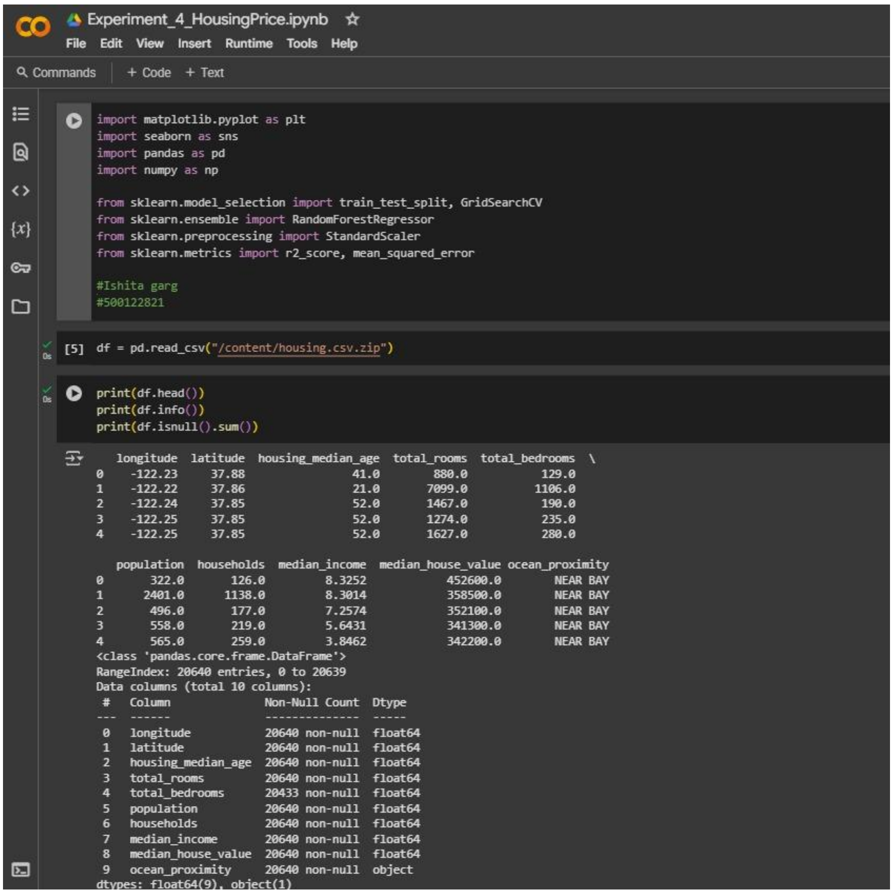


Actual vs Predicted AirQuality (Random Forest)
Predicted AirQuality
Actual AirQuality
0
20
40
60
80
100

Experiment: 4-5

Model Development & Hyper Parameter Tuning for Regression Problems

- *Housing Dataset*


Experiment_4_HousingPrice.ipynb

File Edit View Insert Runtime Tools Help

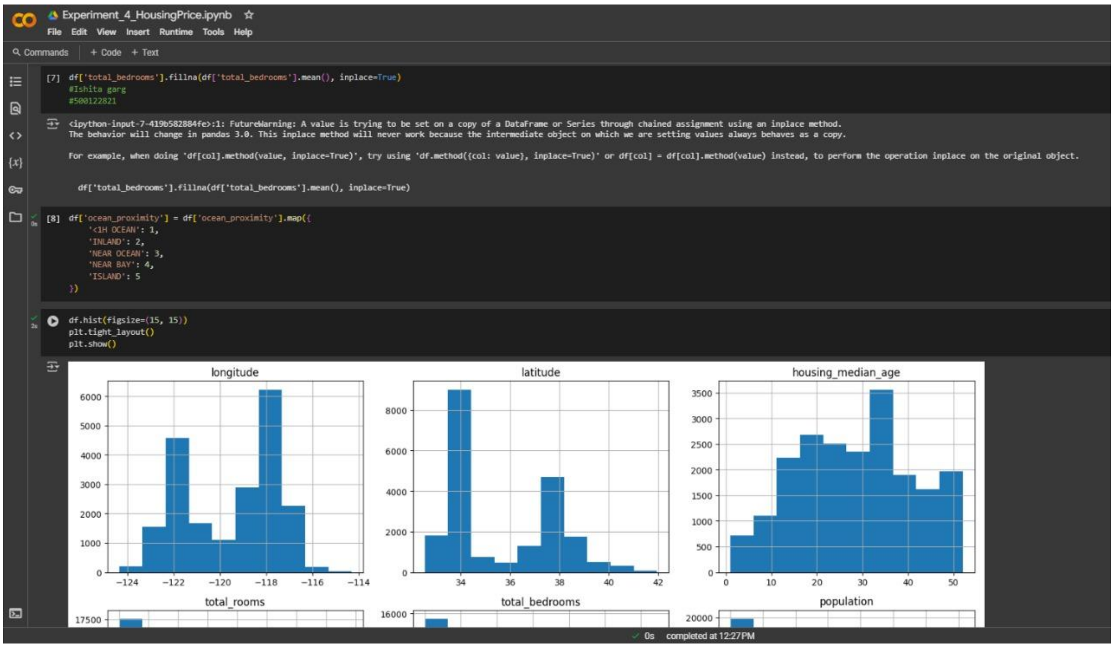
Commands + Code + Text

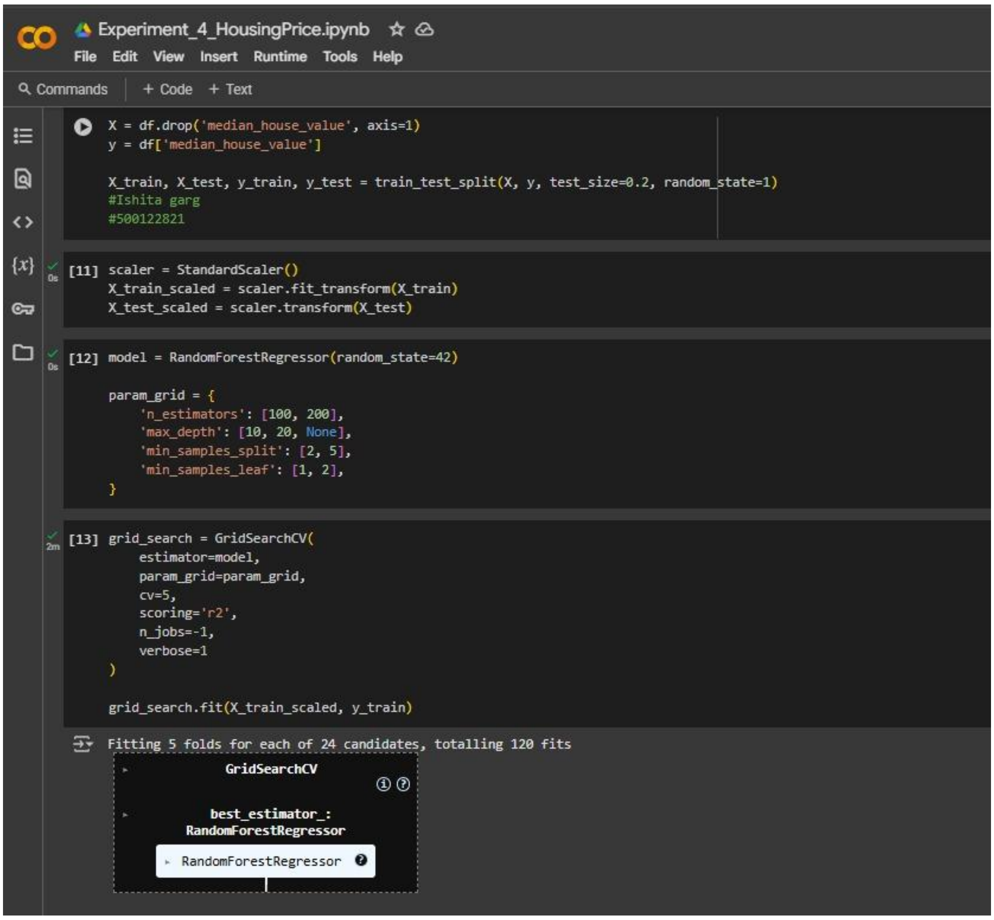
```
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import numpy as np

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import r2_score, mean_squared_error

#Ishita garg
#500122821
```

```
[5] df = pd.read_csv("/content/housing.csv.zip")
```

```
print(df.head())
print(df.info())
print(df.isnull().sum())
```

```
   longitude  latitude  housing_median_age  total_rooms  total_bedrooms  \
0    -122.23     37.88                41.0        880.0           129.0
1    -122.22     37.86                21.0       7099.0          1106.0
2    -122.24     37.85                52.0       1467.0           190.0
3    -122.25     37.85                52.0       1274.0           235.0
4    -122.25     37.85                52.0       1627.0           280.0

   population  households  median_income  median_house_value ocean_proximity
0       322.0       126.0         8.3252            452600.0        NEAR BAY
1      2401.0      1138.0         8.3014            358500.0        NEAR BAY
2       496.0       177.0         7.2574            352100.0        NEAR BAY
3       558.0       219.0         5.6431            341300.0        NEAR BAY
4       565.0       259.0         3.8462            342200.0        NEAR BAY
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20640 entries, 0 to 20639
Data columns (total 10 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   longitude           20640 non-null  float64
 1   latitude            20640 non-null  float64
 2   housing_median_age  20640 non-null  float64
 3   total_rooms         20640 non-null  float64
 4   total_bedrooms      20433 non-null  float64
 5   population          20640 non-null  float64
 6   households          20640 non-null  float64
 7   median_income       20640 non-null  float64
 8   median_house_value  20640 non-null  float64
 9   ocean_proximity     20640 non-null  object
dtypes: float64(9), object(1)
```


Experiment_4_HousingPrice.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
[7] df['total_bedrooms'].fillna(df['total_bedrooms'].mean(), inplace=True)
#Ishita garg
#500122821
<ipython-input-7-419b582884fe>:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.
df['total_bedrooms'].fillna(df['total_bedrooms'].mean(), inplace=True)
[8] df['ocean_proximity'] = df['ocean_proximity'].map({
'<1H OCEAN': 1,
'INLAND': 2,
'NEAR OCEAN': 3,
'NEAR BAY': 4,
'ISLAND': 5
})
df.hist(figsize=(15, 15))
plt.tight_layout()
plt.show()
longitude
latitude
housing_median_age
total_rooms
total_bedrooms
population
completed at 12:27PM


Experiment_4_HousingPrice.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
X = df.drop('median_house_value', axis=1)
y = df['median_house_value']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
#Ishita garg
#500122821
[11] scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
[12] model = RandomForestRegressor(random_state=42)
param_grid = {
'n_estimators': [100, 200],
'max_depth': [10, 20, None],
'min_samples_split': [2, 5],
'min_samples_leaf': [1, 2],
}
[13] grid_search = GridSearchCV(
estimator=model,
param_grid=param_grid,
cv=5,
scoring='r2',
n_jobs=-1,
verbose=1
)
grid_search.fit(X_train_scaled, y_train)
Fitting 5 folds for each of 24 candidates, totalling 120 fits
GridSearchCV
best_estimator_:
RandomForestRegressor
RandomForestRegressor

```
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)

print("Best Parameters:", grid_search.best_params_)
print("R² Score:", r2_score(y_test, y_pred))
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
#Ishita garg
#500122821
```

```
Best Parameters: {'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 200}
R² Score: 0.818389925477026
RMSE: 48807.40434097664
```

- ***Employee Dataset***

Experiment_4_Employee.ipynb

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

from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

#Ishita garg
#500122821
```

```
df = pd.read_csv('/content/employee_attrition_dataset.csv')
```

```
print(df.head())
print(df.info())
print(df.isnull().sum())
```

```
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   Employee_ID                     1000 non-null   int64
 1   Age                             1000 non-null   int64
 2   Gender                          1000 non-null   object
 3   Marital_Status                  1000 non-null   object
 4   Department                      1000 non-null   object
 5   Job_Role                        1000 non-null   object
 6   Job_Level                       1000 non-null   int64
 7   Monthly_Income                  1000 non-null   int64
 8   Hourly_Rate                     1000 non-null   int64
 9   Years_at_Company                1000 non-null   int64
 10  Years_in_Current_Role           1000 non-null   int64
 11  Years_Since_Last_Promotion      1000 non-null   int64
 12  Work_Life_Balance               1000 non-null   int64
 13  Job_Satisfaction                1000 non-null   int64
 14  Performance_Rating              1000 non-null   int64
 15  Training_Hours_Last_Year        1000 non-null   int64
 16  Overtime                        1000 non-null   object
 17  Project_Count                   1000 non-null   int64
 18  Average_Hours_Worked_Per_Week   1000 non-null   int64
 19  Absenteeism                     1000 non-null   int64
 20  Work_Environment_Satisfaction   1000 non-null   int64
 21  Relationship_with_Manager       1000 non-null   int64
 22  Job_Involvement                 1000 non-null   int64
 23  Distance_From_Home              1000 non-null   int64
 24  Number_of_Companies_Worked      1000 non-null   int64
 25  Attrition                       1000 non-null   object
```

0s completed at 12:38 PM

Experiment_4_Employee.ipynb

```
for col in df.select_dtypes(include='object').columns:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
#Ishita garg
#500122821
```

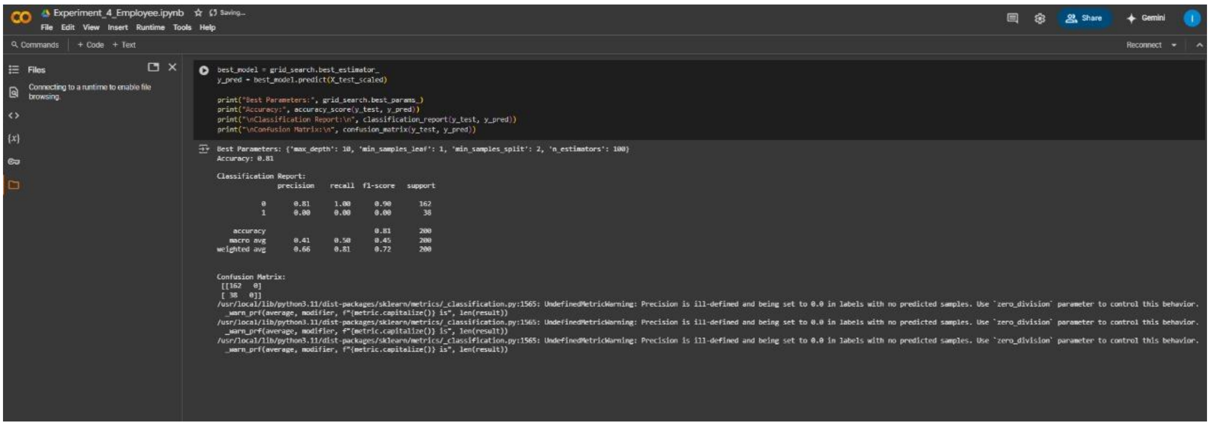
```
X = df.drop('Attrition', axis=1)
y = df['Attrition']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42, stratify=y)
```

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

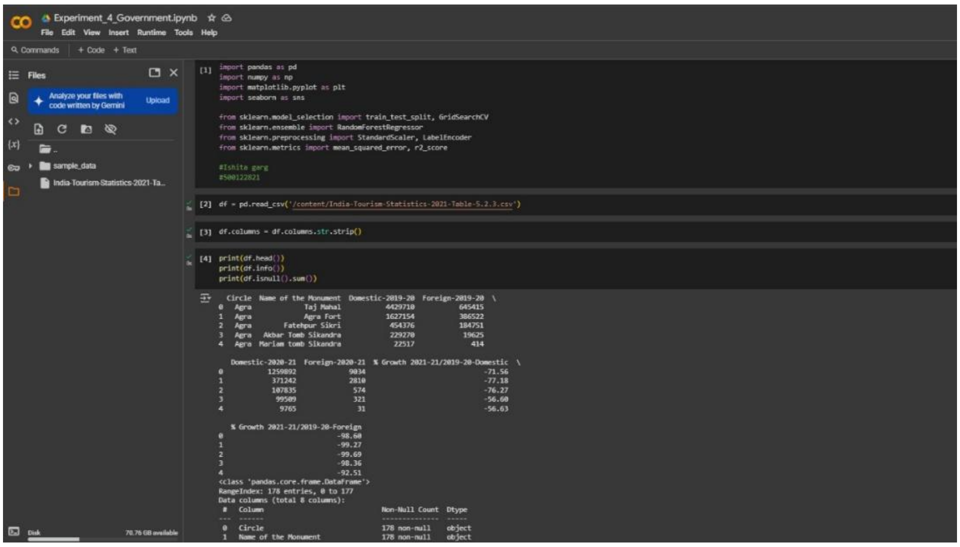
```
model = RandomForestClassifier(random_state=42)

param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
}
```

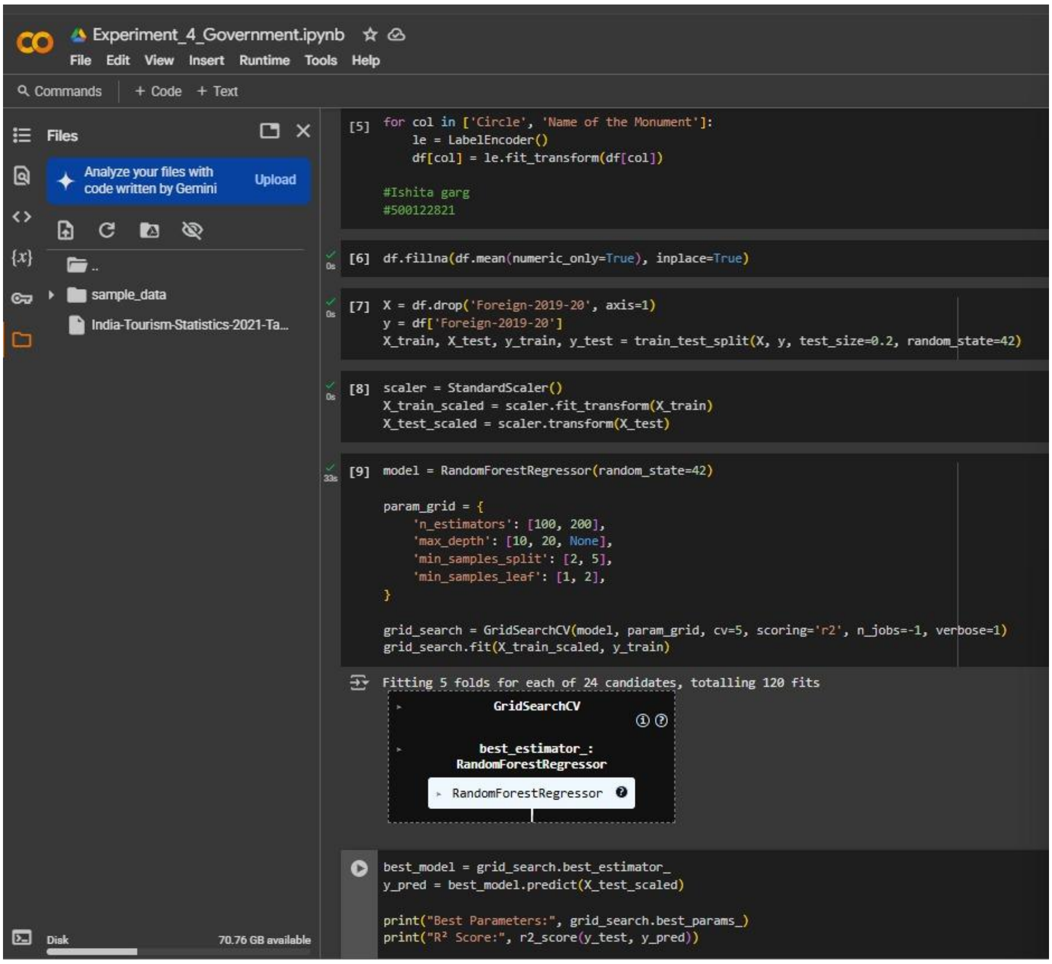
```
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='accuracy', n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)
```

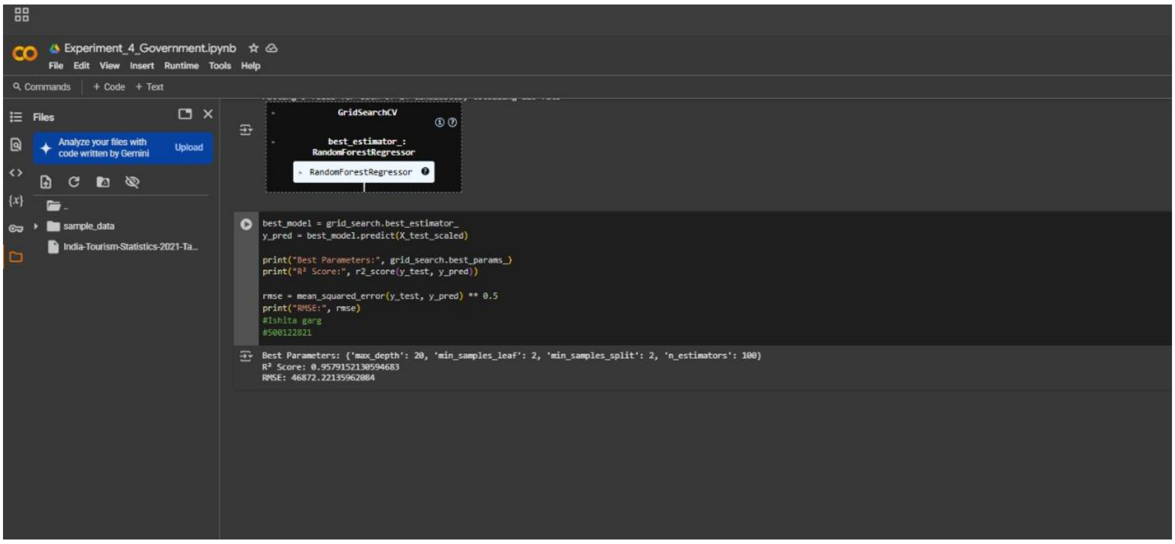
```
Fitting 5 folds for each of 24 candidates, totalling 120 fits
GridSearchCV
best_estimator_:
RandomForestClassifier
RandomForestClassifier
```



- <u>*Government Dataset*</u>




Experiment_4_Government.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Files
Analyze your files with code written by Gemini
Upload
sample_data
India-Tourism-Statistics-2021-Ta...
Disk 70.76 GB available
[5] for col in ['Circle', 'Name of the Monument']:
    le = LabelEncoder()
    df[col] = le.fit_transform(df[col])
#Ishita garg
#500122821
[6] df.fillna(df.mean(numeric_only=True), inplace=True)
[7] X = df.drop('Foreign-2019-20', axis=1)
y = df['Foreign-2019-20']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
[8] scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
[9] model = RandomForestRegressor(random_state=42)
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [10, 20, None],
    'min_samples_split': [2, 5],
    'min_samples_leaf': [1, 2],
}
grid_search = GridSearchCV(model, param_grid, cv=5, scoring='r2', n_jobs=-1, verbose=1)
grid_search.fit(X_train_scaled, y_train)
Fitting 5 folds for each of 24 candidates, totalling 120 fits
GridSearchCV
best_estimator_:
RandomForestRegressor
RandomForestRegressor
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)
print("Best Parameters:", grid_search.best_params_)
print("R² Score:", r2_score(y_test, y_pred))


Experiment_4_Government.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Files
Analyze your files with code written by Gemini
Upload
sample_data
India-Tourism-Statistics-2021-Ta...
GridSearchCV
best_estimator_:
RandomForestRegressor
RandomForestRegressor
best_model = grid_search.best_estimator_
y_pred = best_model.predict(X_test_scaled)
print("Best Parameters:", grid_search.best_params_)
print("R² Score:", r2_score(y_test, y_pred))
rmse = mean_squared_error(y_test, y_pred) ** 0.5
print("RMSE:", rmse)
#Ishita garg
#500122821
Best Parameters: {'max_depth': 20, 'min_samples_leaf': 2, 'min_samples_split': 2, 'n_estimators': 100}
R² Score: 0.9579152130594683
RMSE: 46872.22135962084

Experiment: 6-7

Classification Problem

- *Iris Flower Classification*

MI_Lab_28_01_2025.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
#Ishita Garg
#Sap Id = 500122821
```

```python
data = pd.read_csv('/content/IRIS.csv')
#Ishita Garg
#Sap Id = 500122821
```

```python
data.head()
#Ishita Garg
#Sap Id = 500122821
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa

```python
x = data[['sepal_length' , 'sepal_width' , 'petal_length' , 'petal_width']]
y = data['species']
#Ishita Garg
#Sap Id = 500122821
```

```python
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2)
#Ishita Garg
#Sap Id = 500122821
```

```python
model = DecisionTreeClassifier()
model.fit(x_train, y_train)
#Ishita Garg
#Sap Id = 500122821
```

```
DecisionTreeClassifier
DecisionTreeClassifier()
```

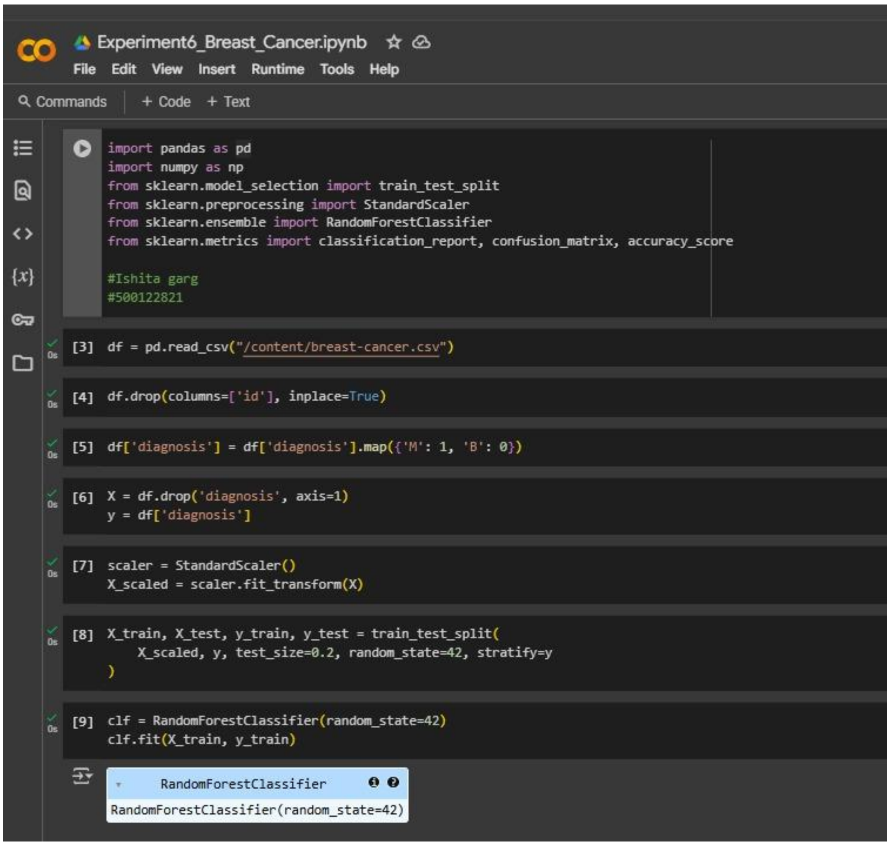
```python
y_pred = model.predict(x_test)
accuracy = accuracy_score(y_test, y_pred) #random state is used to do hit & trial approach and it influences the accuracy
#Ishita Garg
#Sap Id = 500122821
```

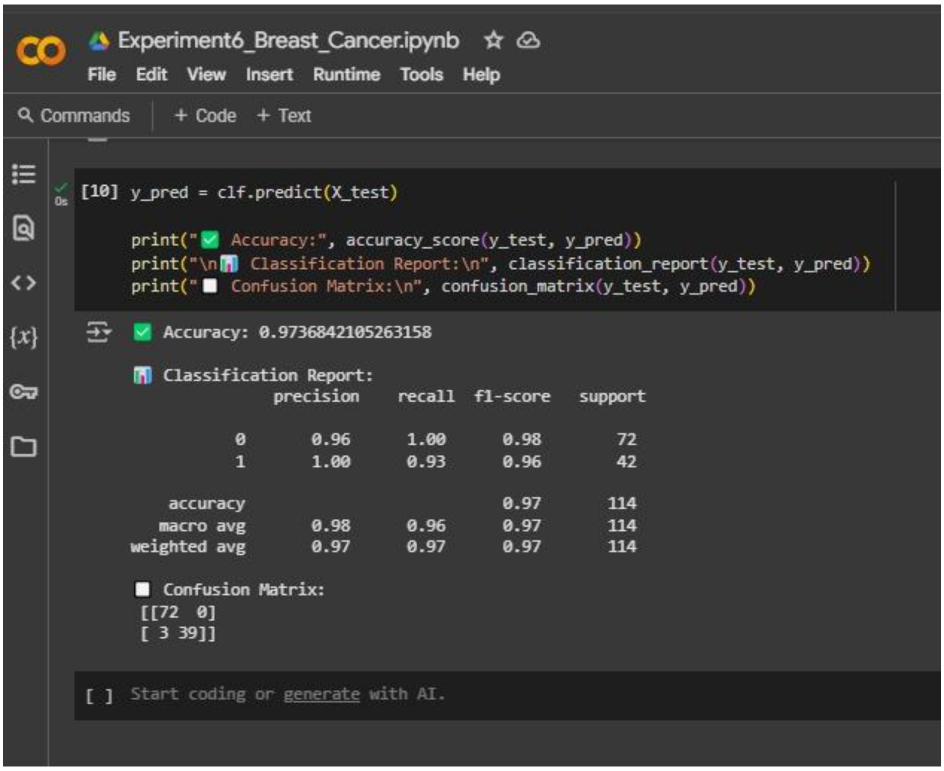
```python
print("Model Accuracy:", accuracy*100 ,"%")
#Ishita Garg
#Sap Id = 500122821
```

```
Model Accuracy: 96.66666666666667 %
```

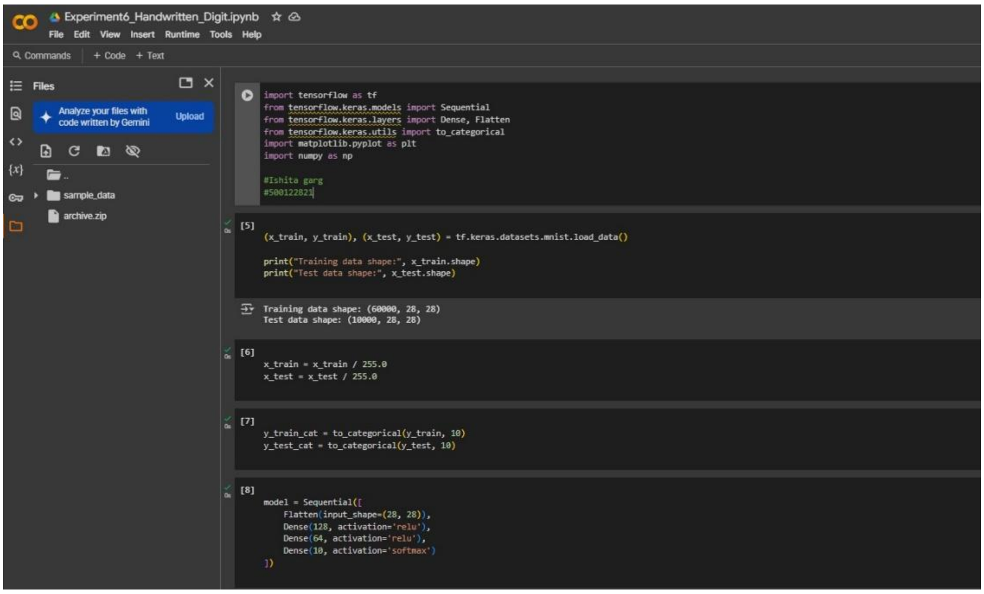
```
Start coding or generate with AI.
```

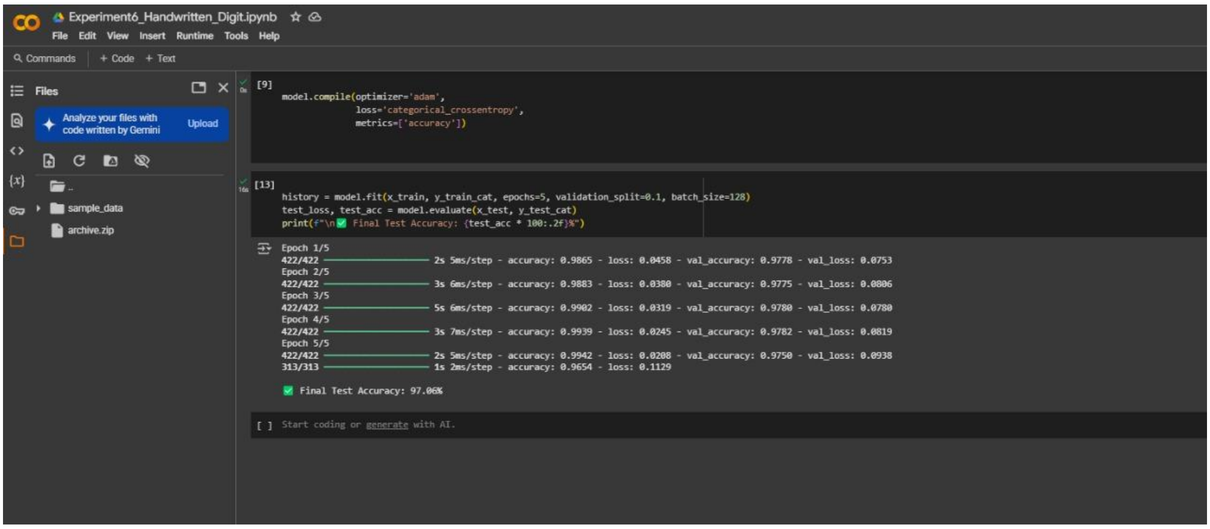
- ***<u>Breast Cancer Diagnosis</u>***



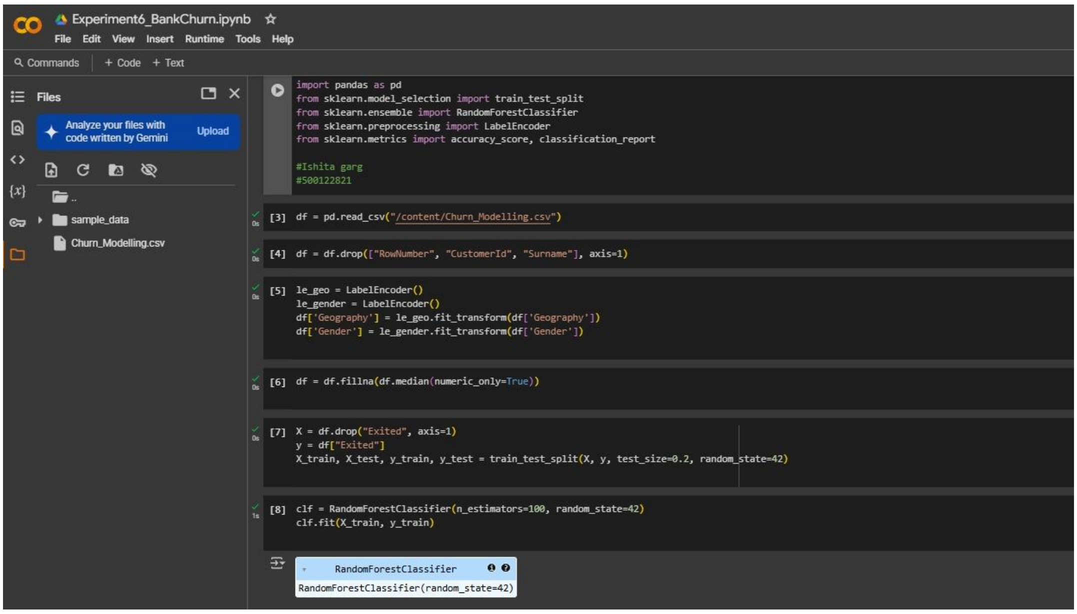


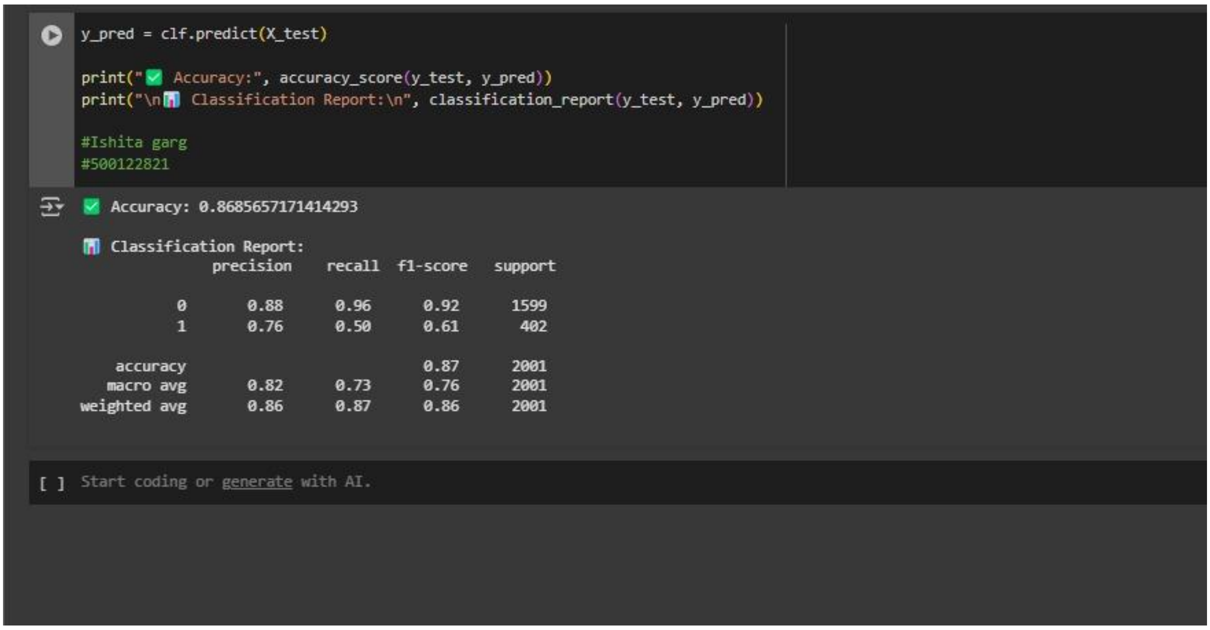
- ***<u>Handwritten Digit Recognition – MNIST</u>***





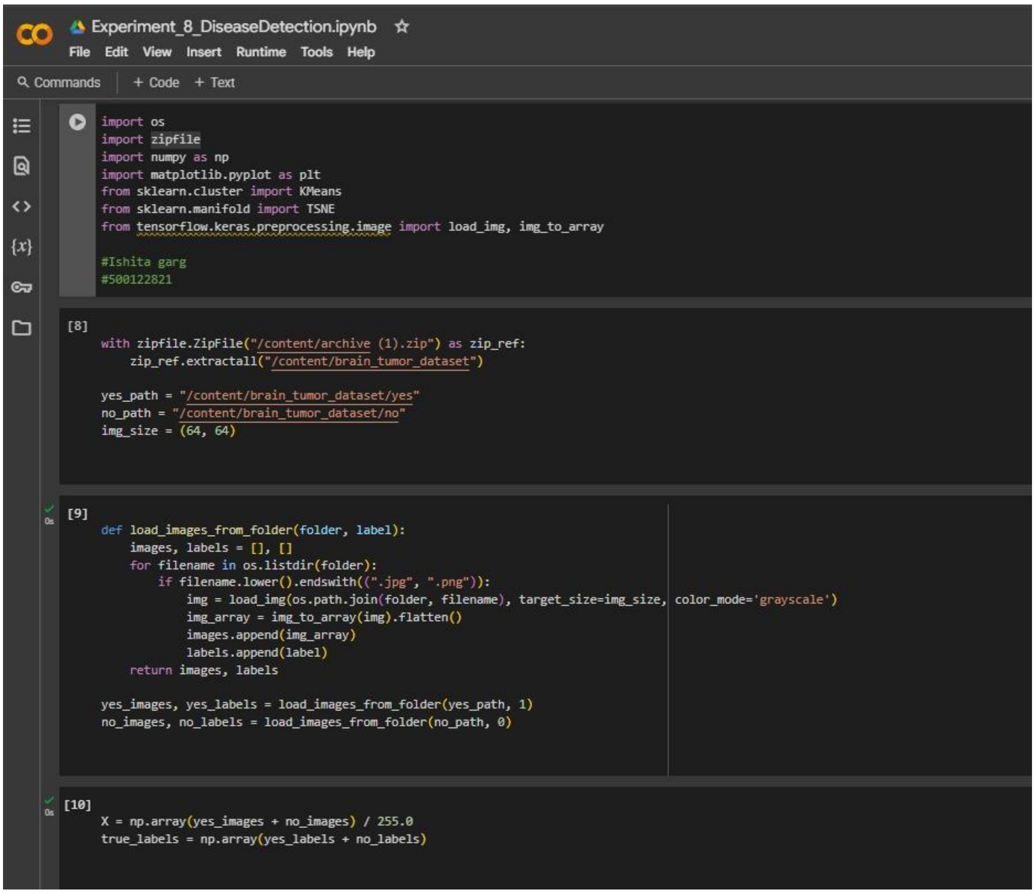
- ***Bank Churn Dataset***





Experiment: 8-9
Clustering Problems

- *Disease Diagnosis from Medical Images*



Experiment_8_DiseaseDetection.ipynb

File Edit View Insert Runtime Tools Help

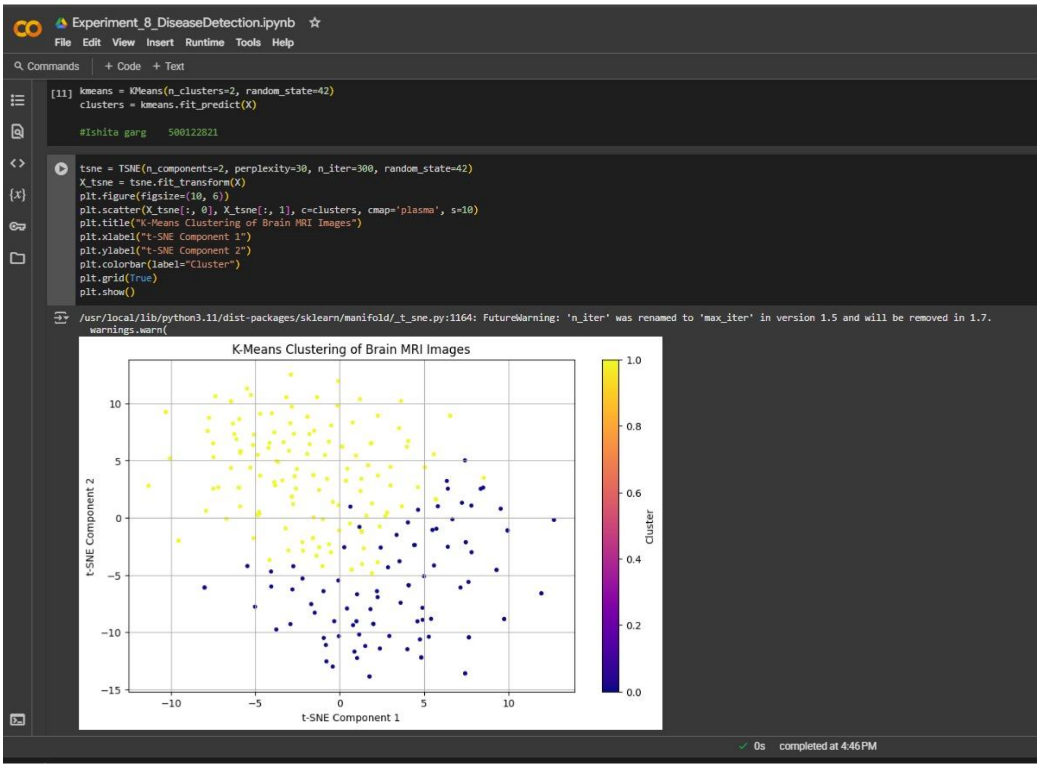
Commands + Code + Text

```python
import os
import zipfile
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.manifold import TSNE
from tensorflow.keras.preprocessing.image import load_img, img_to_array

#Ishita garg
#500122821
```

[8]
```python
with zipfile.ZipFile("/content/archive (1).zip") as zip_ref:
    zip_ref.extractall("/content/brain_tumor_dataset")

yes_path = "/content/brain_tumor_dataset/yes"
no_path = "/content/brain_tumor_dataset/no"
img_size = (64, 64)
```

[9]
```python
def load_images_from_folder(folder, label):
    images, labels = [], []
    for filename in os.listdir(folder):
        if filename.lower().endswith((".jpg", ".png")):
            img = load_img(os.path.join(folder, filename), target_size=img_size, color_mode='grayscale')
            img_array = img_to_array(img).flatten()
            images.append(img_array)
            labels.append(label)
    return images, labels

yes_images, yes_labels = load_images_from_folder(yes_path, 1)
no_images, no_labels = load_images_from_folder(no_path, 0)
```

[10]
```python
X = np.array(yes_images + no_images) / 255.0
true_labels = np.array(yes_labels + no_labels)
```



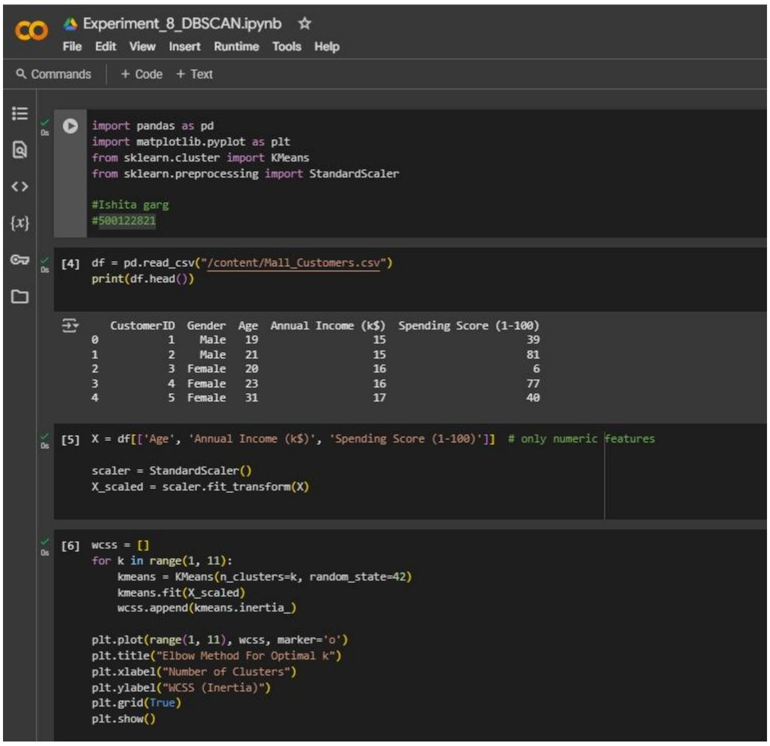
Experiment_8_DiseaseDetection.ipynb

```
[11] kmeans = KMeans(n_clusters=2, random_state=42)
     clusters = kmeans.fit_predict(X)

     #Ishita garg    500122821
```

```
tsne = TSNE(n_components=2, perplexity=30, n_iter=300, random_state=42)
X_tsne = tsne.fit_transform(X)
plt.figure(figsize=(10, 6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=clusters, cmap='plasma', s=10)
plt.title("K-Means Clustering of Brain MRI Images")
plt.xlabel("t-SNE Component 1")
plt.ylabel("t-SNE Component 2")
plt.colorbar(label="Cluster")
plt.grid(True)
plt.show()
```

```
/usr/local/lib/python3.11/dist-packages/sklearn/manifold/_t_sne.py:1164: FutureWarning: 'n_iter' was renamed to 'max_iter' in version 1.5 and will be removed in 1.7.
  warnings.warn(
```


- ***DBSCAN***



Experiment_8_DBSCAN.ipynb

```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

#Ishita garg
#500122821
```

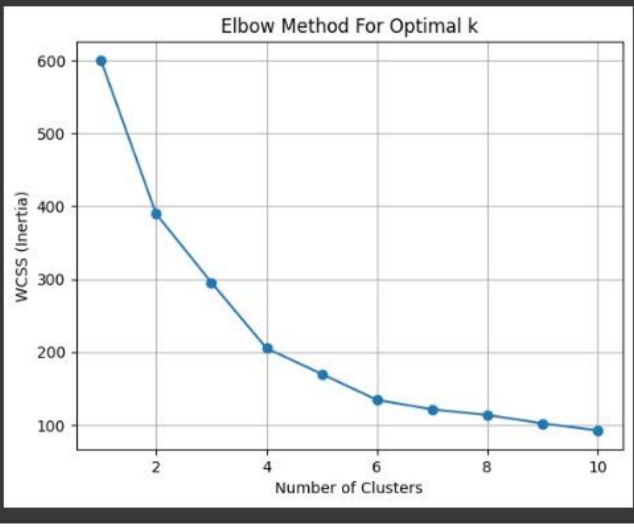
```
[4] df = pd.read_csv("/content/Mall_Customers.csv")
    print(df.head())
```

```
   CustomerID  Gender  Age  Annual Income (k$)  Spending Score (1-100)
0           1    Male   19                  15                      39
1           2    Male   21                  15                      81
2           3  Female   20                  16                       6
3           4  Female   23                  16                      77
4           5  Female   31                  17                      40
```

```
[5] X = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]  # only numeric features

    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
```

```
[6] wcss = []
    for k in range(1, 11):
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X_scaled)
        wcss.append(kmeans.inertia_)

    plt.plot(range(1, 11), wcss, marker='o')
    plt.title("Elbow Method For Optimal k")
    plt.xlabel("Number of Clusters")
    plt.ylabel("WCSS (Inertia)")
    plt.grid(True)
    plt.show()
```

```python
wcss = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

plt.plot(range(1, 11), wcss, marker='o')
plt.title("Elbow Method For Optimal k")
plt.xlabel("Number of Clusters")
plt.ylabel("WCSS (Inertia)")
plt.grid(True)
plt.show()

#Ishita garg
#500122821
```



Experiment_8_DBSCAN.ipynb

File Edit View Insert Runtime Tools Help

Commands + Code + Text

```python
kmeans = KMeans(n_clusters=5, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

plt.figure(figsize=(8, 6))
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'],
            c=df['Cluster'], cmap='viridis', s=60)
plt.title("Customer Segments Based on Income and Spending")
plt.xlabel("Annual Income (k$)")
plt.ylabel("Spending Score (1-100)")
plt.grid(True)
plt.colorbar(label="Cluster")
plt.show()

#Ishita garg
#500122821
```