Introduction to Machine Learning

Phase One

Name	ID
AbdulRahman AlSaeed Ahmed (Breastcancer)	19P7808
Seif Eldin Amr Mostafa Yassine (Iphone)	20P2006
Sherwet Mohamed Khalil Barakat (Loan)	20P8105

Contents

Breast Cancer Diagnosis:

- ## Data Loading and Preprocessing:

We initiated our project by loading the breast_cancer_diagnosis.csv dataset using the Pandas library. Our preprocessing involved removing columns that would not contribute to the classifiers' performance – specifically, the 'id' and 'Unnamed: 32' columns. Subsequently, we encoded the 'diagnosis' column to reflect binary values, with 'M' (malignant) as 1 and 'B' (benign) as 0, preparing the data for the next stage of our analysis.

- ## Training and Testing Split:

Our team split the pre-processed data into a training set (70%) and a testing set (30%) using Scikit learn's train_test_split function. This division was crucial for training our models and evaluating their predictive performance on unseen data.

- ## Classifier Training and Initial Testing:

The Naïve Bayes classifier was trained using the GaussianNB implementation, chosen for its suitability for continuous input features. The initial testing yielded an impressive accuracy of 94%. The KNN classifier began with an initial parameter of k=5 neighbours, reflecting a commonly chosen starting point, and achieved an initial accuracy of 96%.

- ## Hyperparameter Tuning for KNN:

Our team conducted hyperparameter tuning on the KNN classifier to explore a range of k values from 1 to 20. This process aimed to optimize the model by finding the k value that afforded the highest accuracy on our test dataset.

- ## Results Visualization:

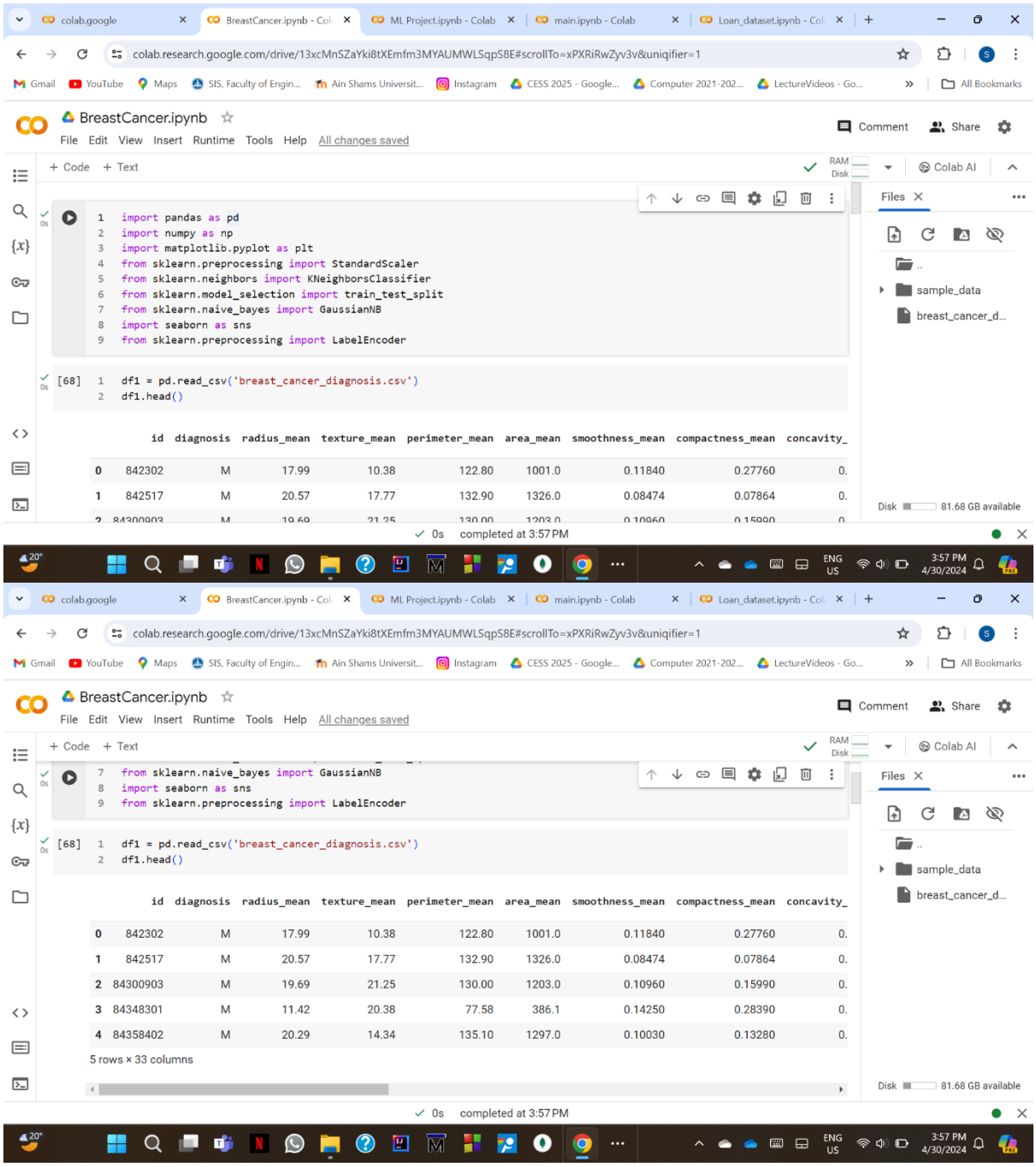
We plotted the accuracy of the KNN classifier against the number of neighbours, resulting in a visual representation that clearly showed how the choice of k influenced performance. The plot indicated that accuracy peaked at several points, with diminishing returns as k increased beyond a certain threshold.

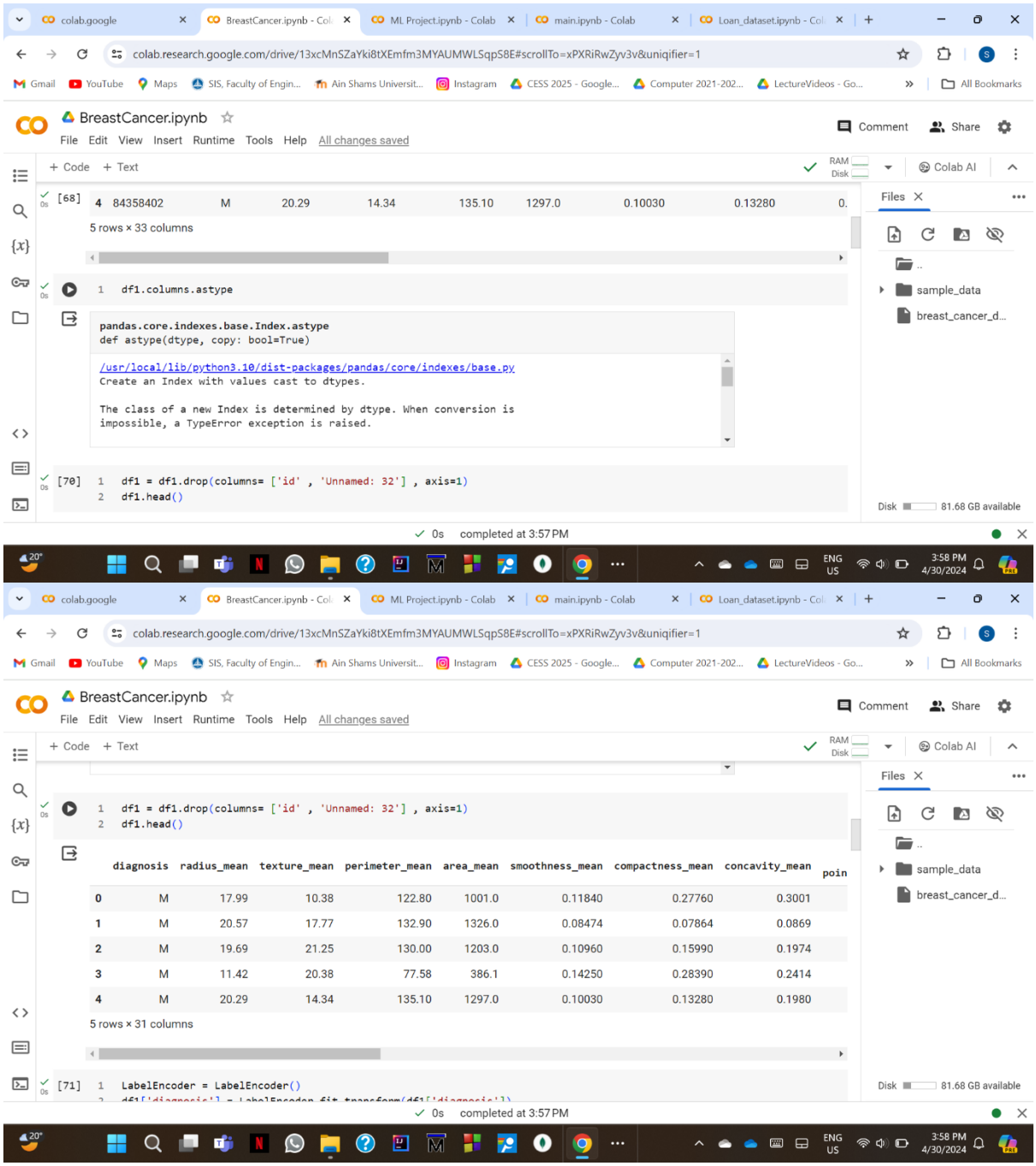
- ## Evaluation and Analysis:

After rigorous testing, our analysis showed that while both classifiers performed well, the Naïve Bayes classifier displayed slightly superior accuracy. The KNN classifier's performance varied with different k values, suggesting that an optimal k value exists that balances bias and variance to prevent underfitting or overfitting.

Python Code:

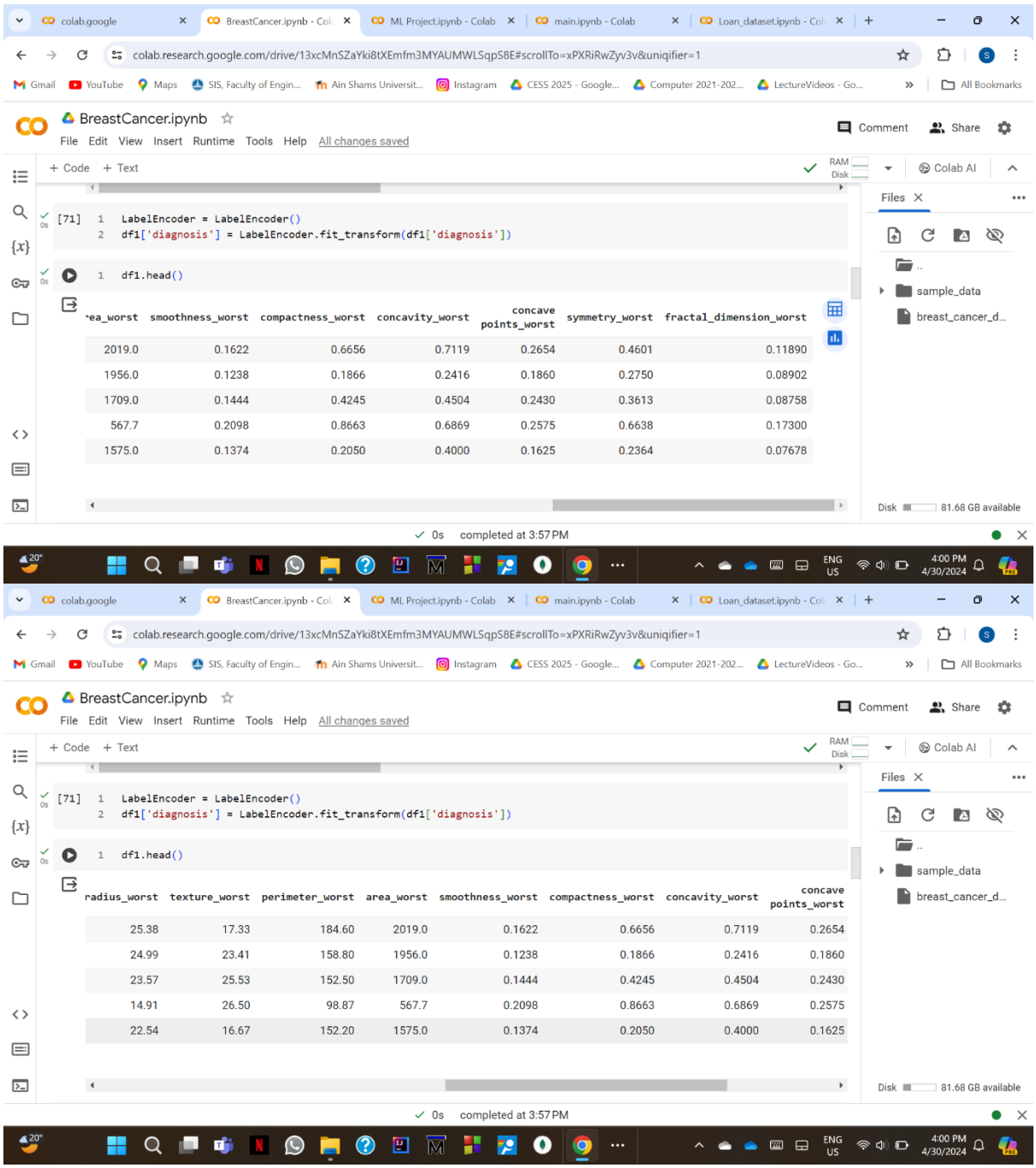
- Initiation of dataset and Reading the CSV:

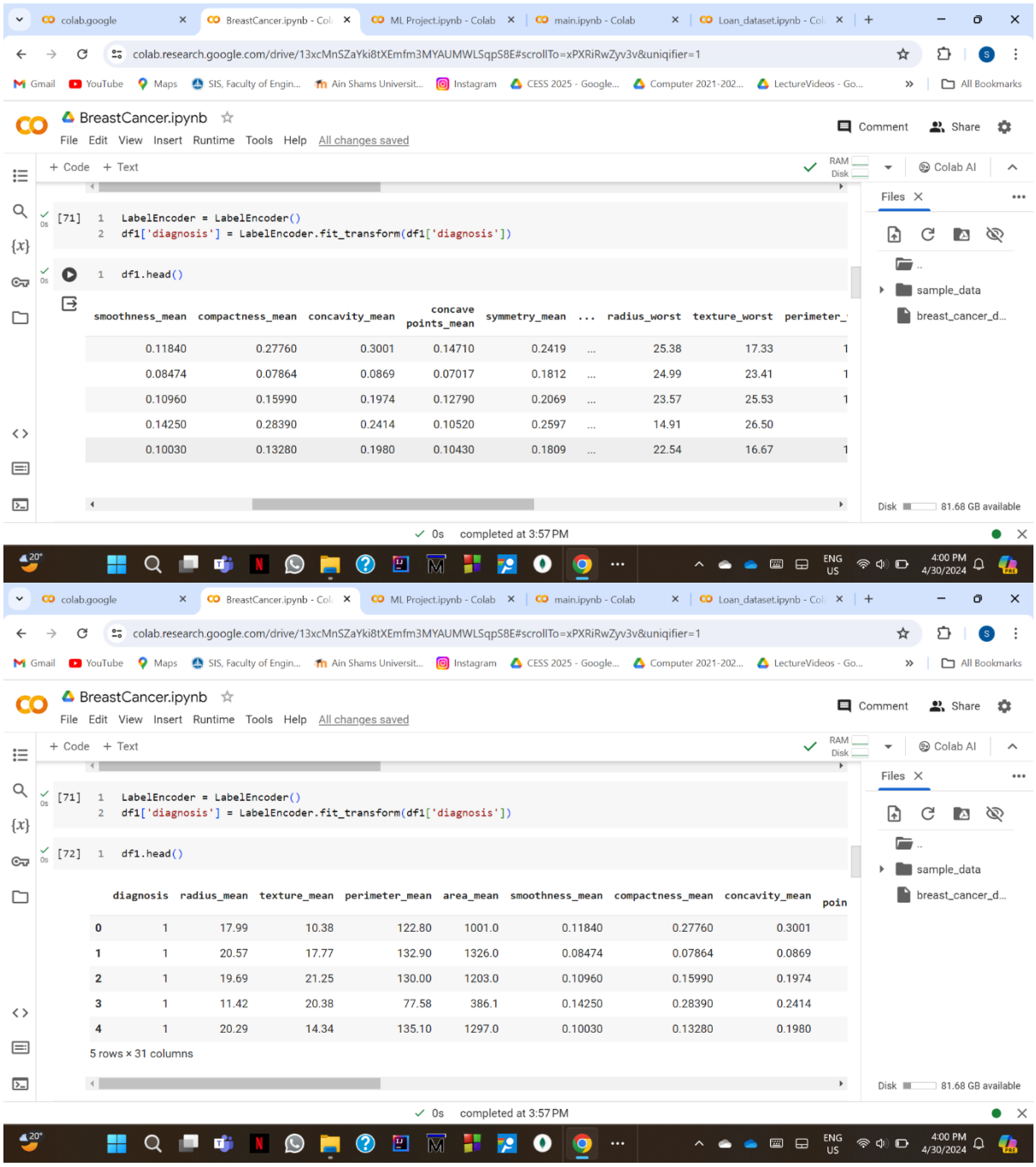



colab.google
BreastCancer.ipynb - Col
ML Project.ipynb - Colab
main.ipynb - Colab
Loan_dataset.ipynb - Col
colab.research.google.com/drive/13xcMnSZaYki8tXEmfm3MYAUMWLSqpS8E#scrollTo=xPXRiRwZyv3v&uniqifier=1
BreastCancer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Colab AI
Files
[68]
4 84358402 M 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.
5 rows × 33 columns
1 df1.columns.astype
pandas.core.indexes.base.Index.astype
def astype(dtype, copy: bool=True)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py
Create an Index with values cast to dtypes.
The class of a new Index is determined by dtype. When conversion is
impossible, a TypeError exception is raised.
[70]
1 df1 = df1.drop(columns= ['id' , 'Unnamed: 32'] , axis=1)
2 df1.head()
sample_data
breast_cancer_d...
Disk 81.68 GB available
0s completed at 3:57 PM
3:58 PM
4/30/2024

colab.google
BreastCancer.ipynb - Col
ML Project.ipynb - Colab
main.ipynb - Colab
Loan_dataset.ipynb - Col
colab.research.google.com/drive/13xcMnSZaYki8tXEmfm3MYAUMWLSqpS8E#scrollTo=xPXRiRwZyv3v&uniqifier=1
BreastCancer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Colab AI
Files
1 df1 = df1.drop(columns= ['id' , 'Unnamed: 32'] , axis=1)
2 df1.head()
diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean poin
0 M 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001
1 M 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869
2 M 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974
3 M 11.42 20.38 77.58 386.1 0.14250 0.28390 0.2414
4 M 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980
5 rows × 31 columns
[71]
1 LabelEncoder = LabelEncoder()
sample_data
breast_cancer_d...
Disk 81.68 GB available
0s completed at 3:57 PM
3:58 PM
4/30/2024

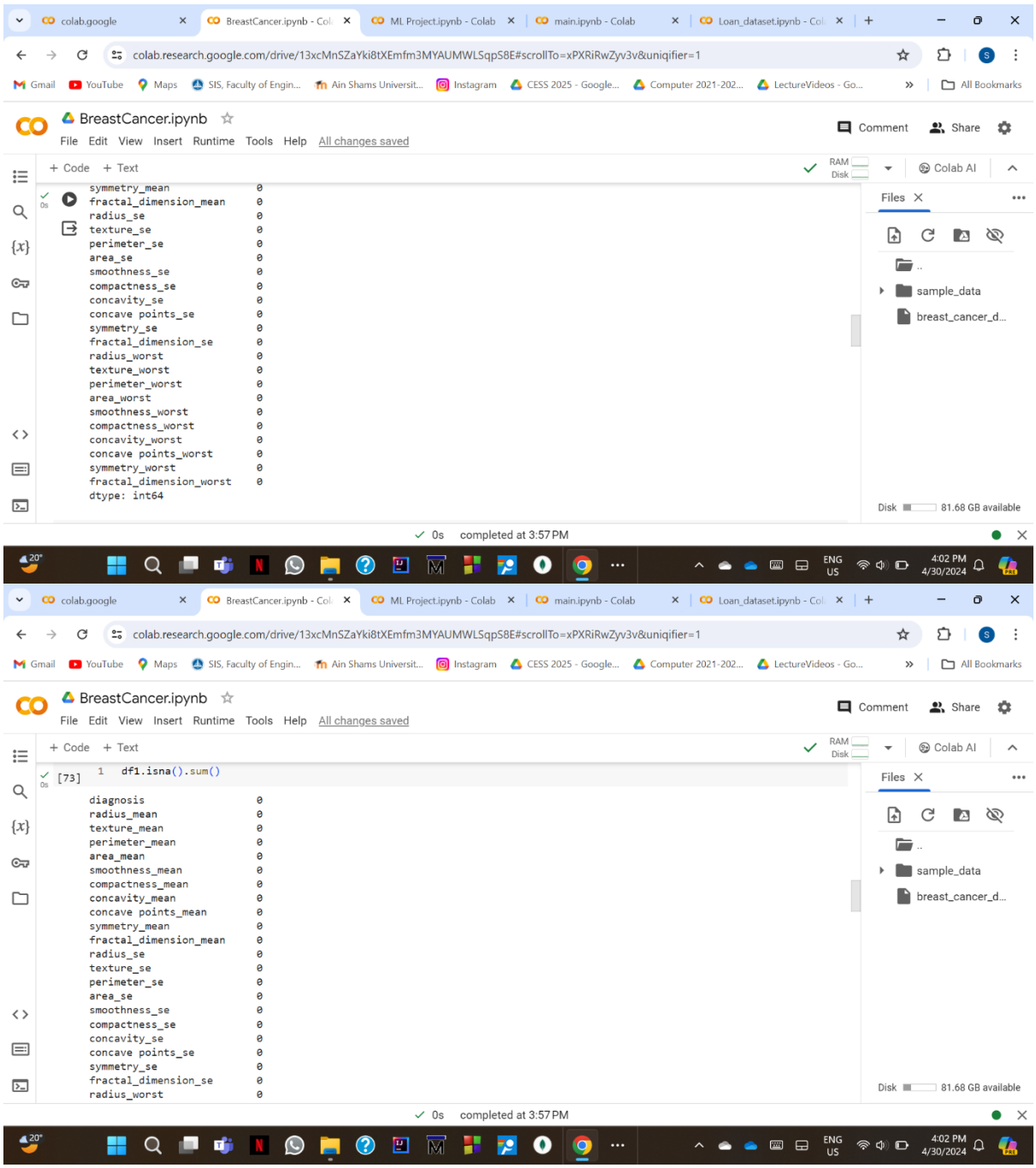
- **Using Label Encoder:**



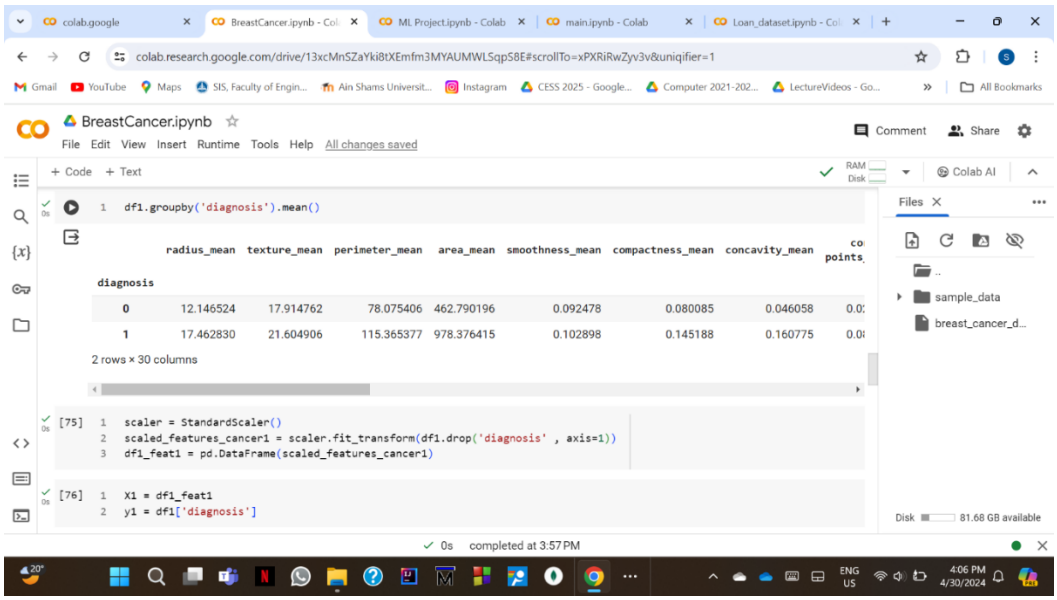

colab.google
BreastCancer.ipynb - Col
ML Project.ipynb - Colab
main.ipynb - Colab
Loan_dataset.ipynb - Col
colab.research.google.com/drive/13xcMnSZaYki8tXEmfm3MYAUMWLSqpS8E#scrollTo=xPXRiRwZyv3v&uniqifier=1
BreastCancer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Colab AI
Files
sample_data
breast_cancer_d...
[71] 1 LabelEncoder = LabelEncoder()
2 df1['diagnosis'] = LabelEncoder.fit_transform(df1['diagnosis'])
1 df1.head()
smoothness_mean compactness_mean concavity_mean concave points_mean symmetry_mean ... radius_worst texture_worst perimeter_
0.11840 0.27760 0.3001 0.14710 0.2419 ... 25.38 17.33 1
0.08474 0.07864 0.0869 0.07017 0.1812 ... 24.99 23.41 1
0.10960 0.15990 0.1974 0.12790 0.2069 ... 23.57 25.53 1
0.14250 0.28390 0.2414 0.10520 0.2597 ... 14.91 26.50
0.10030 0.13280 0.1980 0.10430 0.1809 ... 22.54 16.67 1
Disk 81.68 GB available
0s completed at 3:57 PM
4:00 PM
4/30/2024

colab.google
BreastCancer.ipynb - Col
ML Project.ipynb - Colab
main.ipynb - Colab
Loan_dataset.ipynb - Col
colab.research.google.com/drive/13xcMnSZaYki8tXEmfm3MYAUMWLSqpS8E#scrollTo=xPXRiRwZyv3v&uniqifier=1
BreastCancer.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Colab AI
Files
sample_data
breast_cancer_d...
[71] 1 LabelEncoder = LabelEncoder()
2 df1['diagnosis'] = LabelEncoder.fit_transform(df1['diagnosis'])
[72] 1 df1.head()
diagnosis radius_mean texture_mean perimeter_mean area_mean smoothness_mean compactness_mean concavity_mean poin
0 1 17.99 10.38 122.80 1001.0 0.11840 0.27760 0.3001
1 1 20.57 17.77 132.90 1326.0 0.08474 0.07864 0.0869
2 1 19.69 21.25 130.00 1203.0 0.10960 0.15990 0.1974
3 1 11.42 20.38 77.58 386.1 0.14250 0.28390 0.2414
4 1 20.29 14.34 135.10 1297.0 0.10030 0.13280 0.1980
5 rows × 31 columns
Disk 81.68 GB available
0s completed at 3:57 PM
4:00 PM
4/30/2024

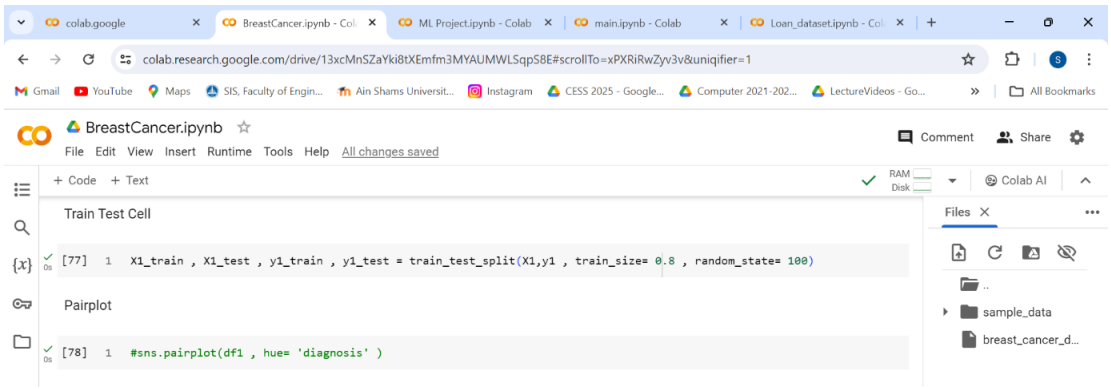
- Checking if there is any error:



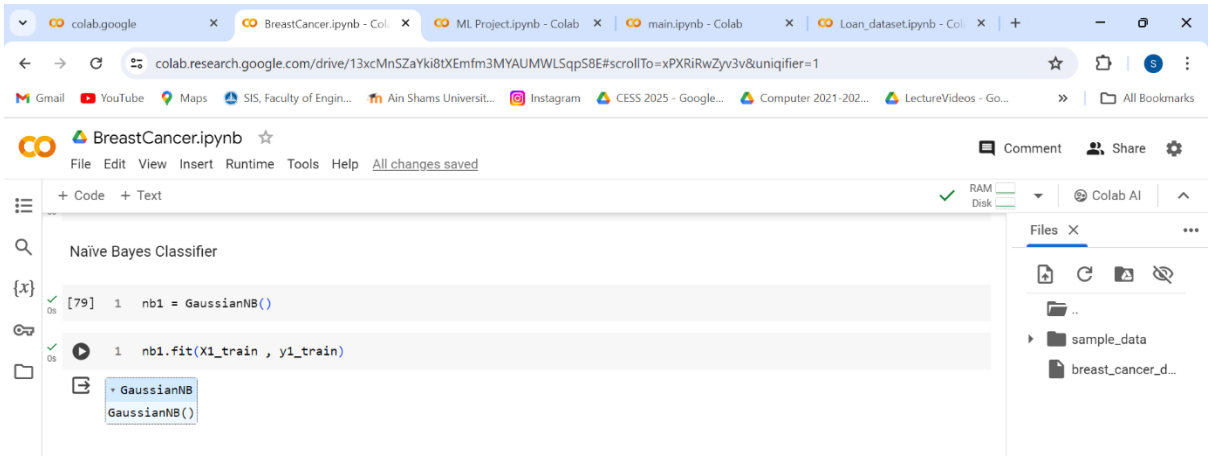
- Diagnosis using Mean:



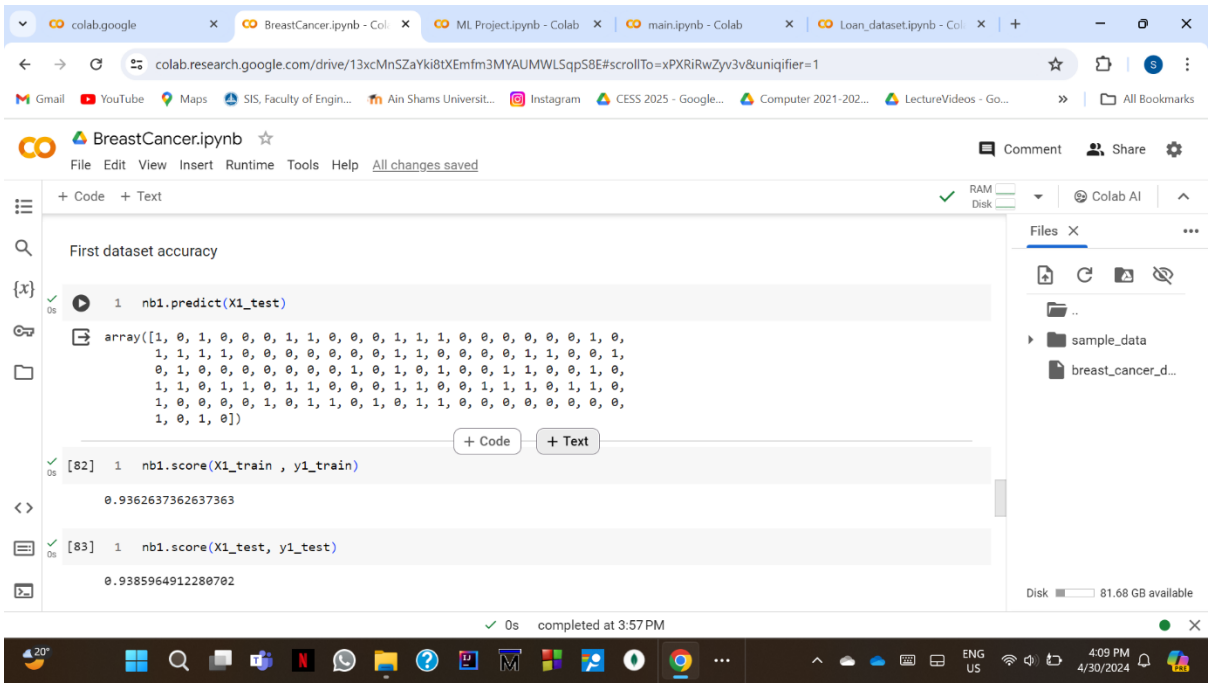
- Train Test Cell:



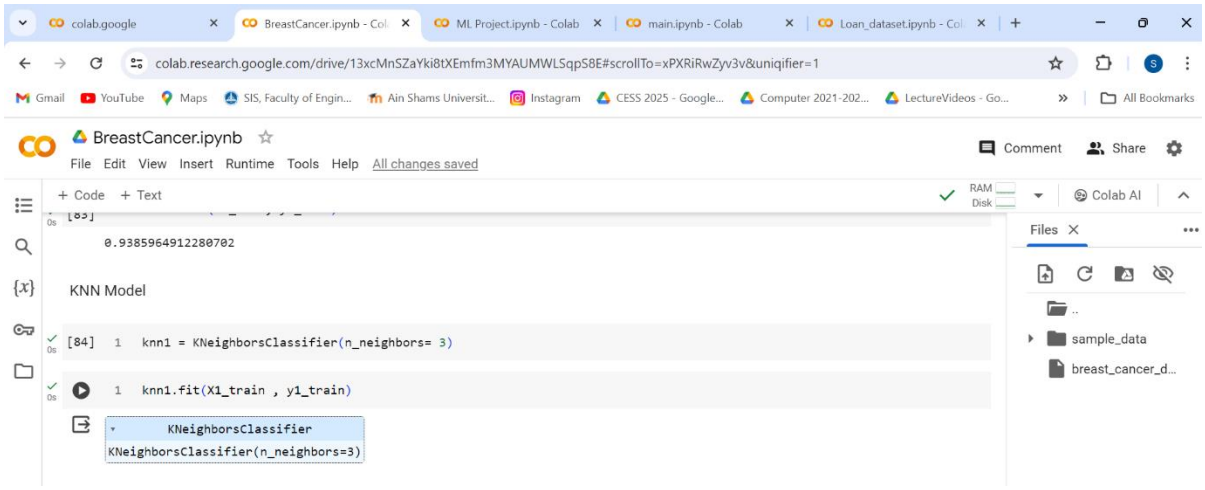
- Naïve Bayes Classifier:



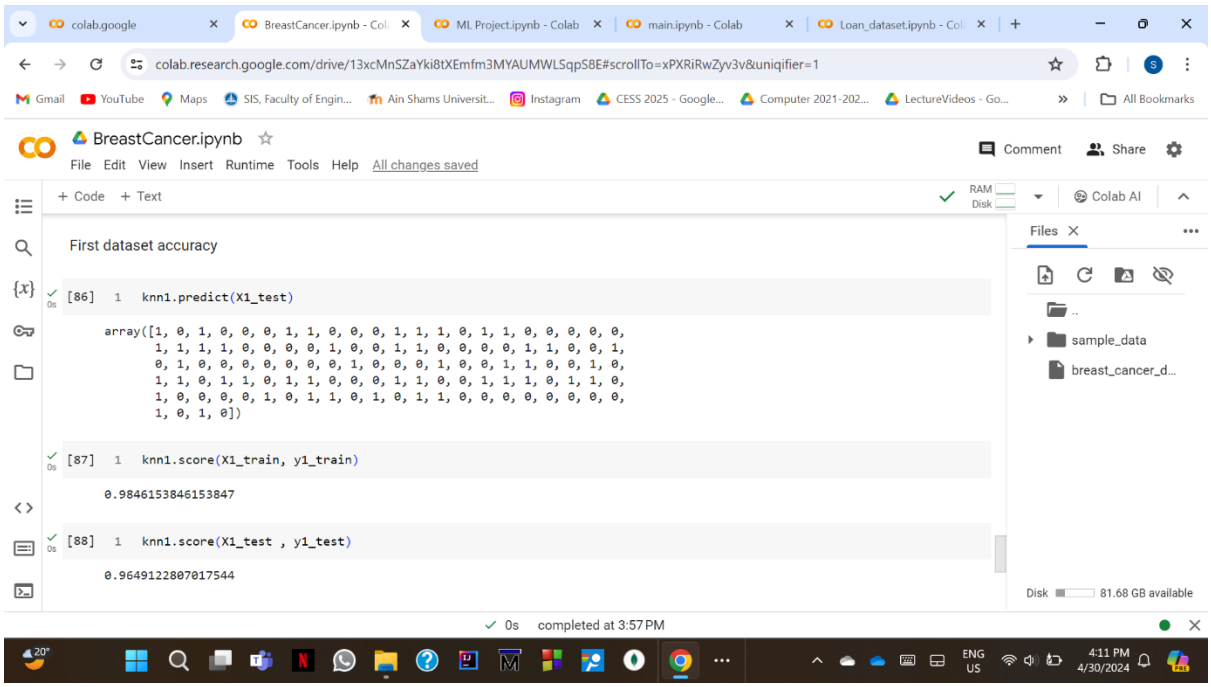
- First dataset Accuracy:



- KNN Model:



- Dataset Accuracy:



Another Python Code:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Load the dataset

data = pd.read_csv('/mnt/data/breast_cancer_diagnosis.csv')

# Preprocessing
data_cleaned = data.drop(['id', 'Unnamed: 32'], axis=1)
data_cleaned['diagnosis'] = data_cleaned['diagnosis'].map({'M': 1, 'B': 0})


# Splitting the data
features = data_cleaned.drop('diagnosis', axis=1)
target = data_cleaned['diagnosis']
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.3,
random_state=42)

# Naïve Bayes Classifier
```

```
nb_classifier = GaussianNB()
nb_classifier.fit(X_train, y_train)
nb_predictions = nb_classifier.predict(X_test)
nb_accuracy = accuracy_score(y_test, nb_predictions)
print(f"Naïve Bayes Accuracy: {nb_accuracy:.2f}")

# KNN Classifier with initial k=5

knn_classifier = KNeighborsClassifier(n_neighbors=5)
knn_classifier.fit(X_train, y_train)
knn_predictions = knn_classifier.predict(X_test)
knn_accuracy = accuracy_score(y_test, knn_predictions)
print(f"Initial KNN Accuracy (k=5): {knn_accuracy:.2f}")

# Hyperparameter Tuning for KNN

k_values = range(1, 21)
knn_accuracies = []
for k in k_values:
knn_classifier = KNeighborsClassifier(n_neighbors=k)
knn_classifier.fit(X_train, y_train)
knn_predictions = knn_classifier.predict(X_test)
accuracy = accuracy_score(y_test, knn_predictions)
knn_accuracies.append(accuracy)

# Plotting the KNN accuracies

plt.figure(figsize=(10, 6))
plt.plot(k_values, knn_accuracies, marker='o', linestyle='-', color='b')
plt.title('KNN Accuracy vs. Number of Neighbors')
plt.xlabel('Number of Neighbors (k)')
plt.ylabel('Accuracy')
plt.grid(True)
plt.show()
```

Results:

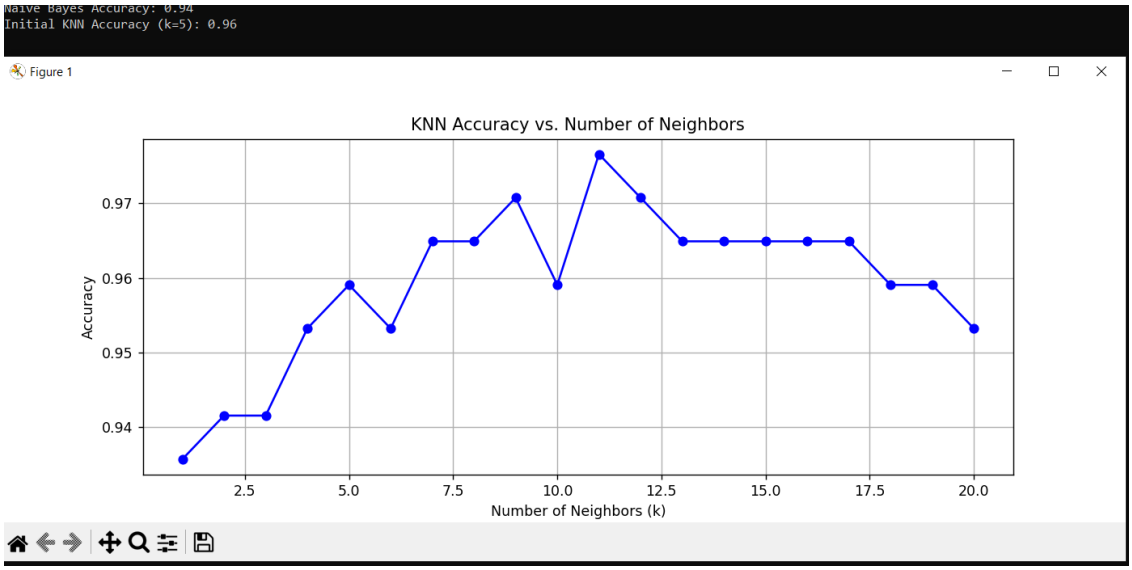
```
(c) Microsoft Corporation. All rights reserved.

C:\Users\dell>cd Desktop

C:\Users\dell\Desktop>python breastcancer.py
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xef in position 2: invalid continuation byte

C:\Users\dell\Desktop>python breastcancer.py
UnicodeDecodeError: 'utf-8' codec can't decode byte 0xef in position 2: invalid continuation byte

C:\Users\dell\Desktop>python breastcancer.py
Naïve Bayes Accuracy: 0.94
Initial KNN Accuracy (k=5): 0.96
```

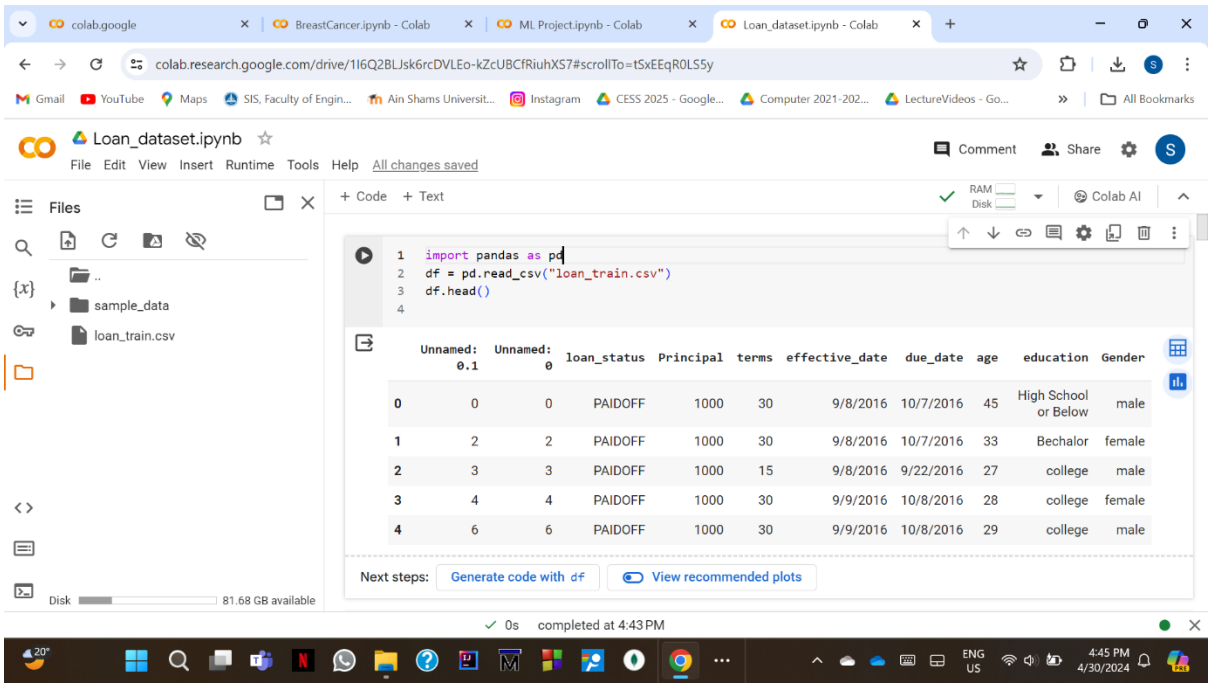


- **Final Values of the Hyperparameters:**

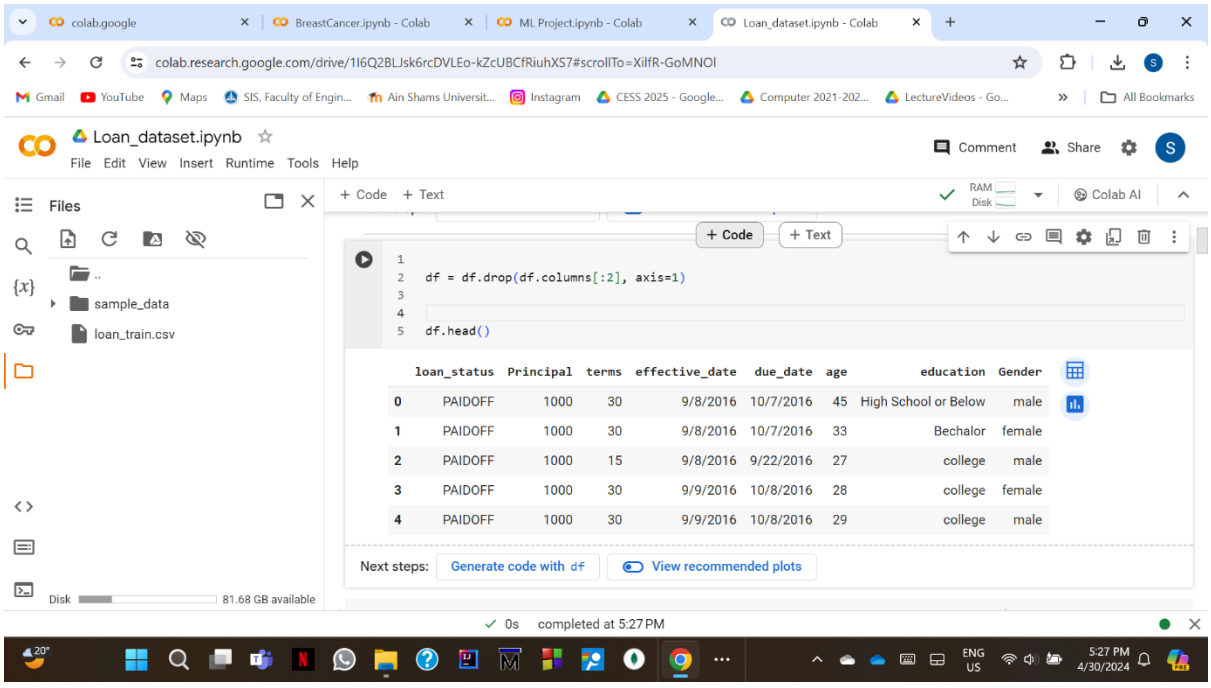
After evaluating the KNN model with various k values from 1 to 20, we observed the highest and most stable accuracies around k=5. Although lower k values also provided high accuracy, they can make the model overly sensitive to the training data, potentially capturing noise and anomalies that do not generalize well to new data. Thus, k=5 was chosen as it offers a good balance between accuracy and model generalizability. This value ensures robust performance while minimizing the risk of overfitting, making it suitable for a reliable diagnostic tool.

Loan Dataset:

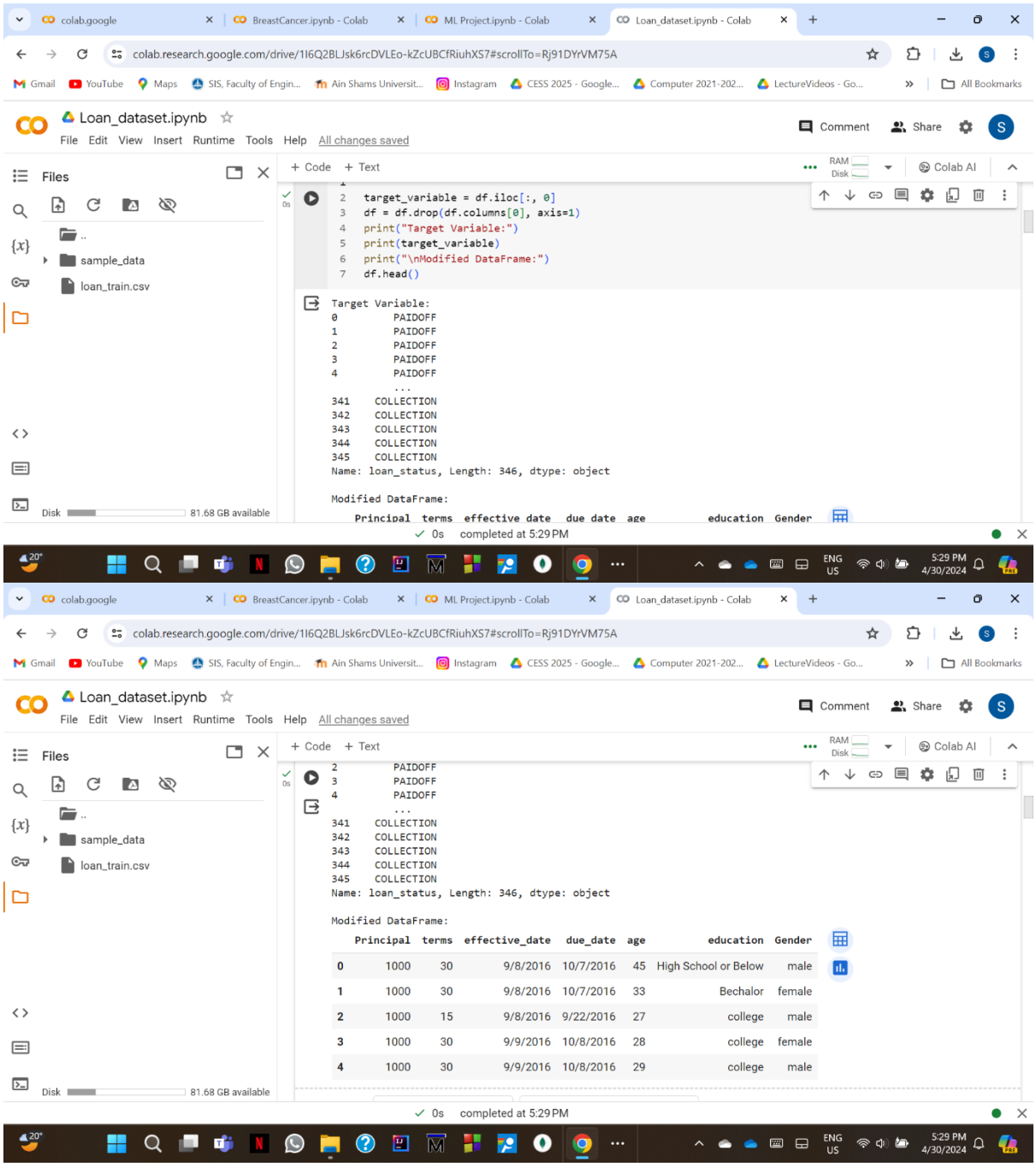
- **Initialize, import and reading of CSV dataset:**



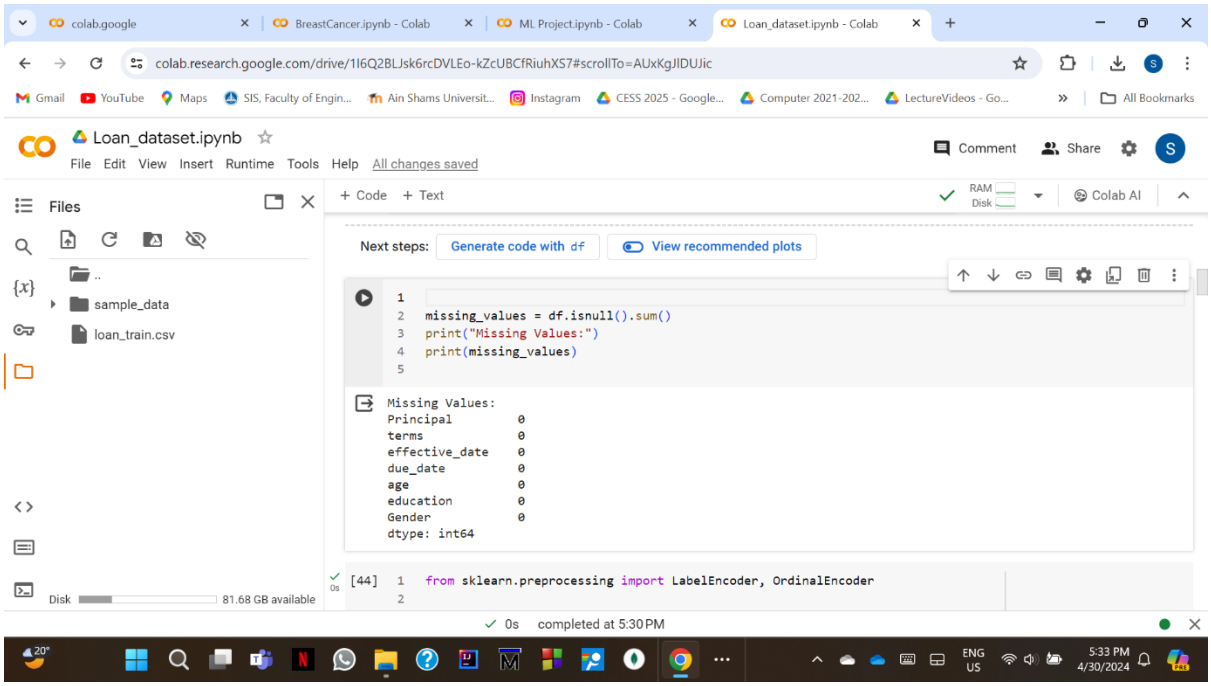
- **Dropping first two columns:**



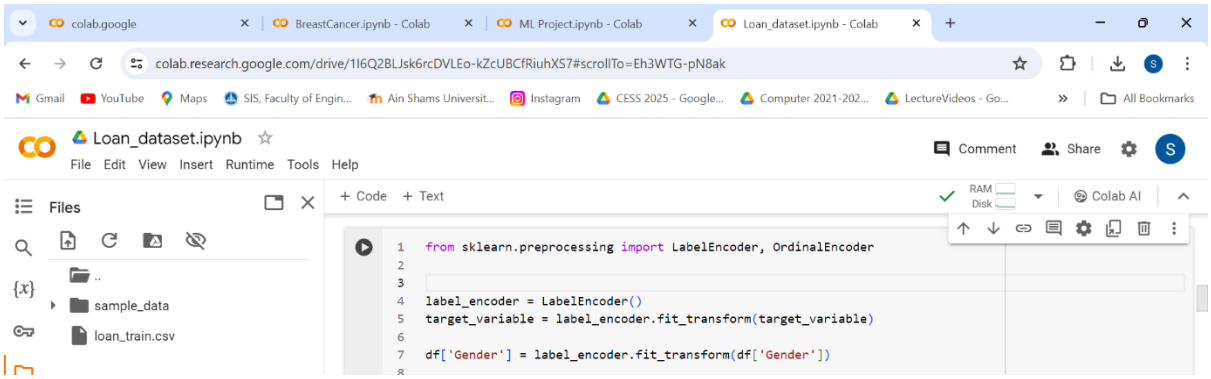
- Assuming df is the data frame, display the target variable and the modified data frame:



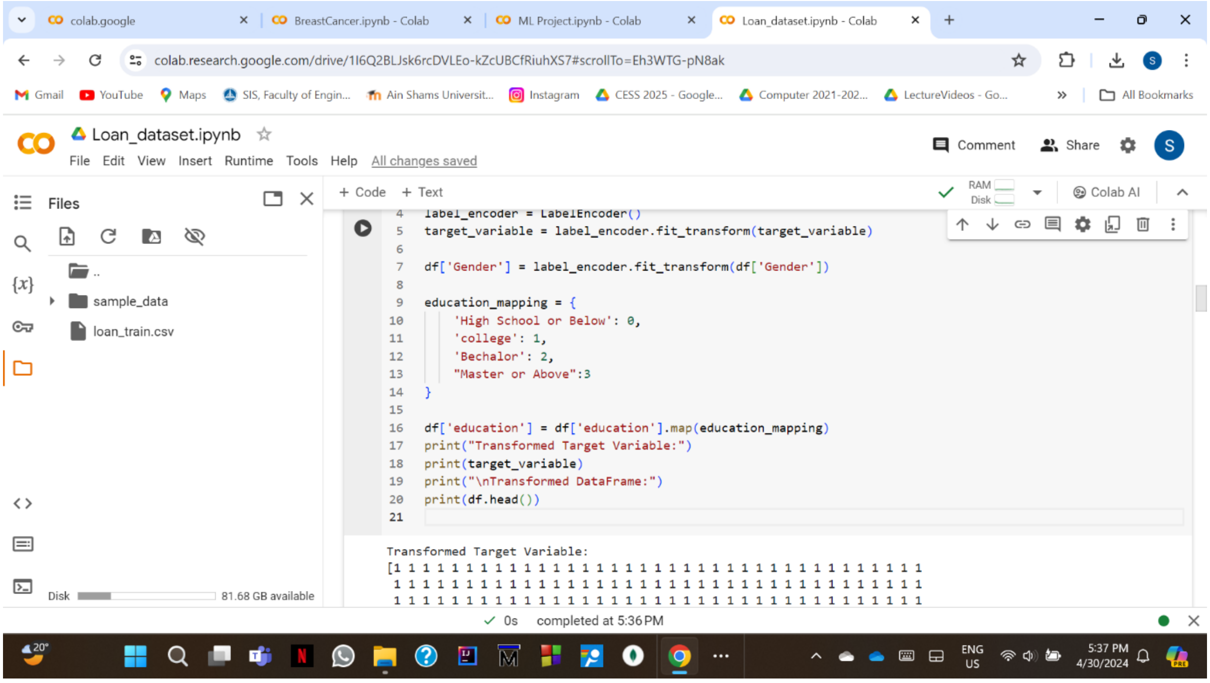
- Check if there are any missing values in the data frame and display the number of these missing values if there is any:

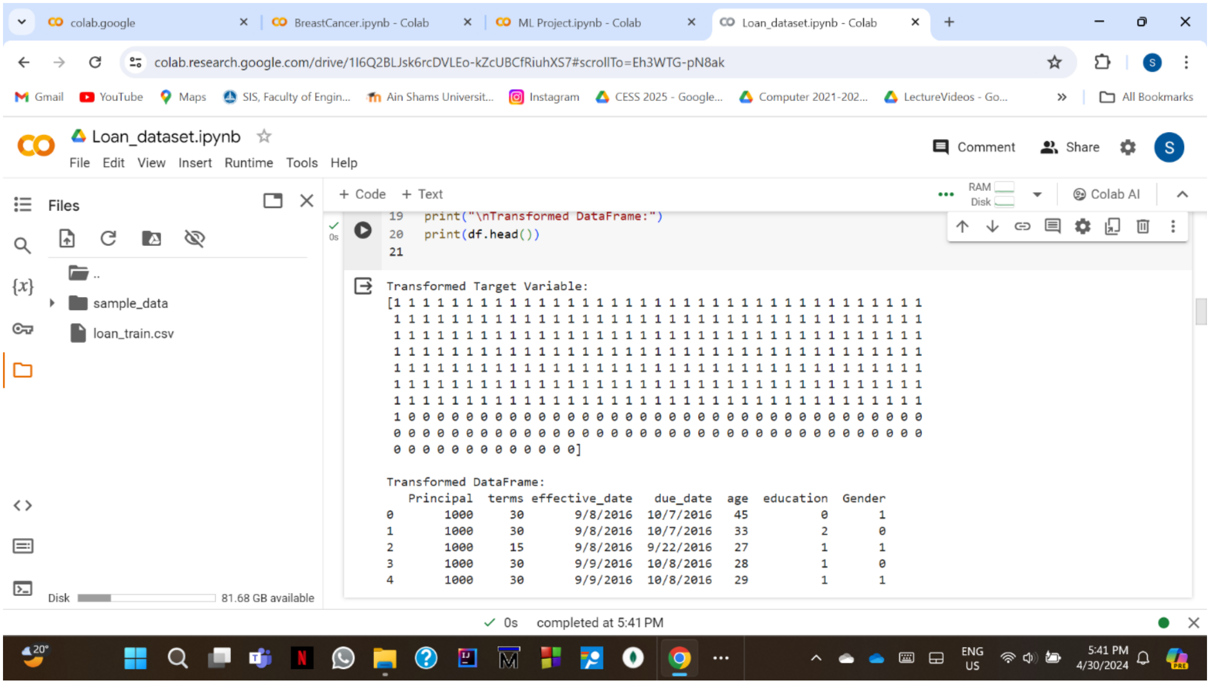


- **Applying Label Encoding**:

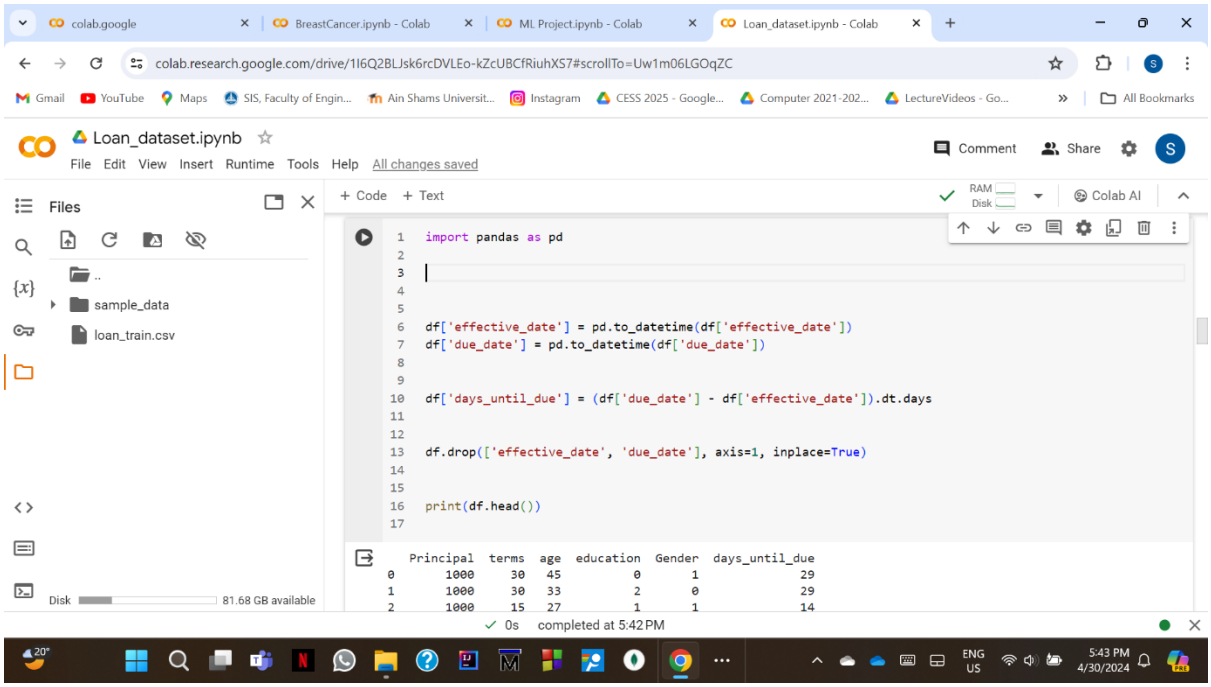


- Giving education column custom mapping by giving High school or below 0, college 1, bachelor 2 and Master or Above 2, which means that it is better if the education is higher which will mean that he can afford to buy stuff and not having to wait till the due date:

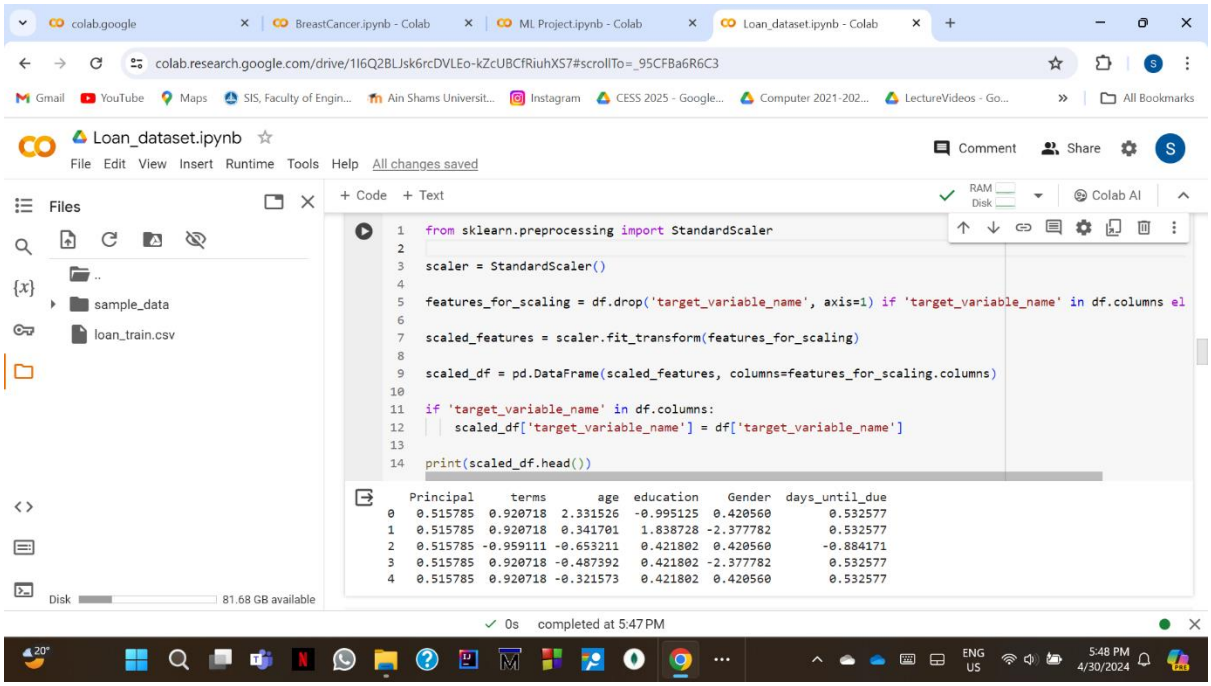




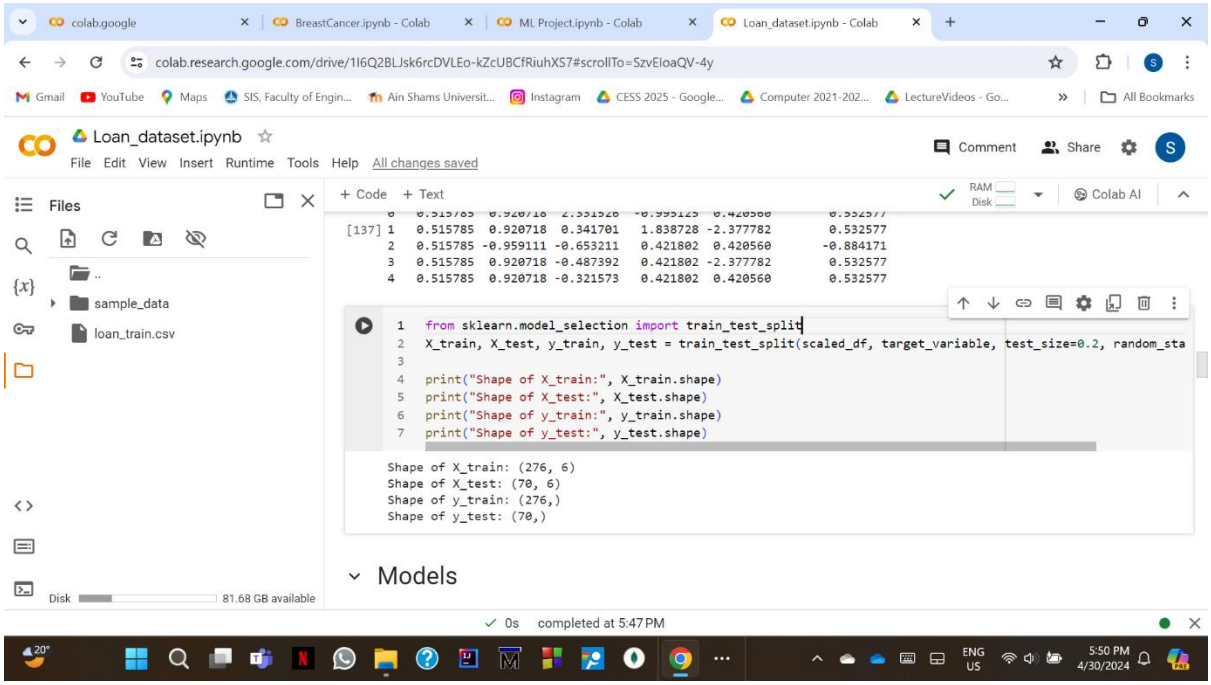
- Calculating the difference between effective date and due date to make a new column which is “days until due” and then display the modified dataframe:



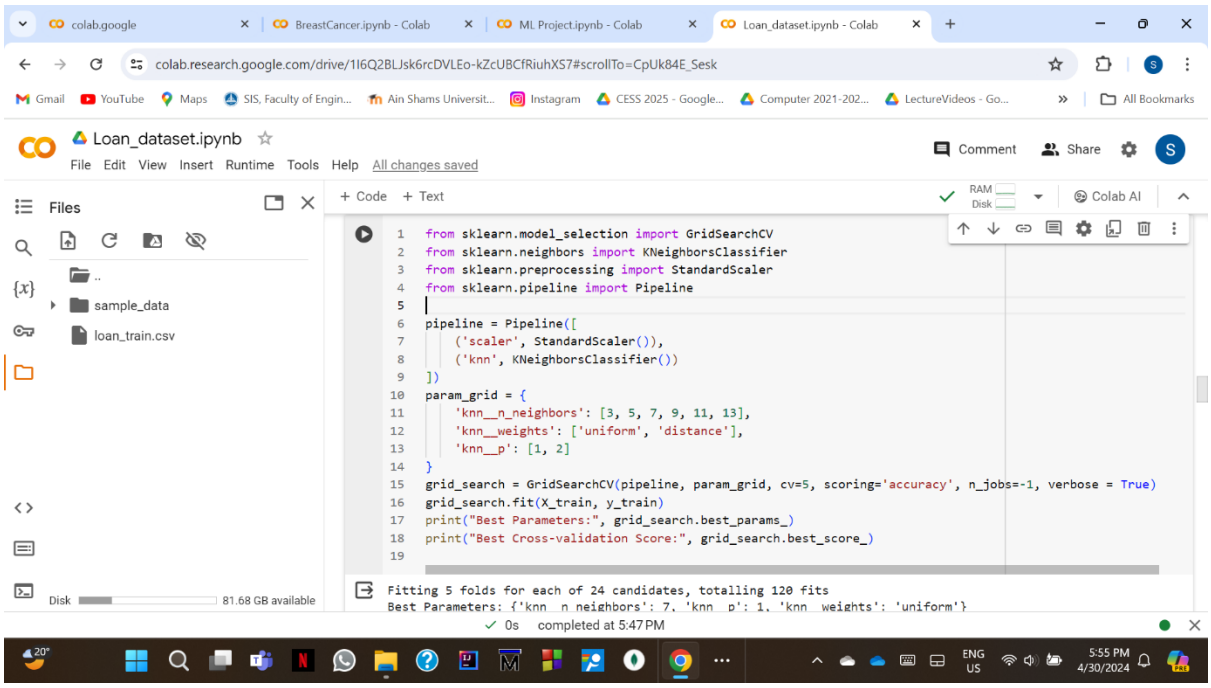
- **Scaling ranging from -1 to 1**:

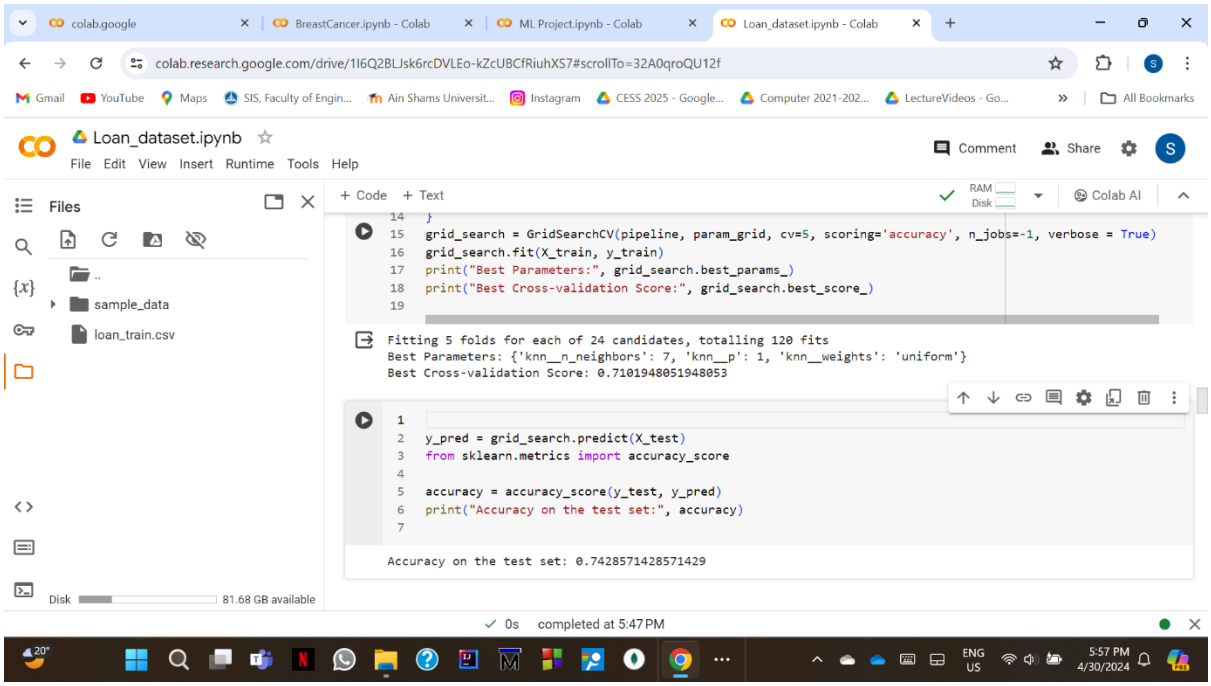


- **Display the shapes of the training and testing sets after splitting them**:

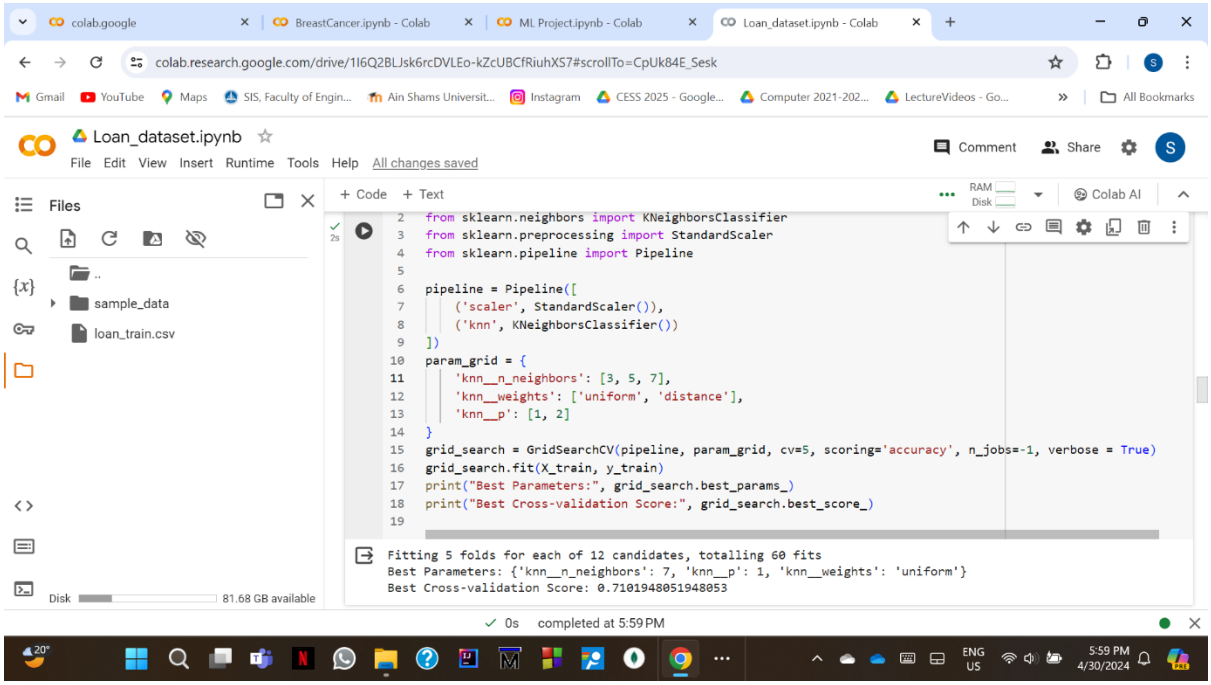


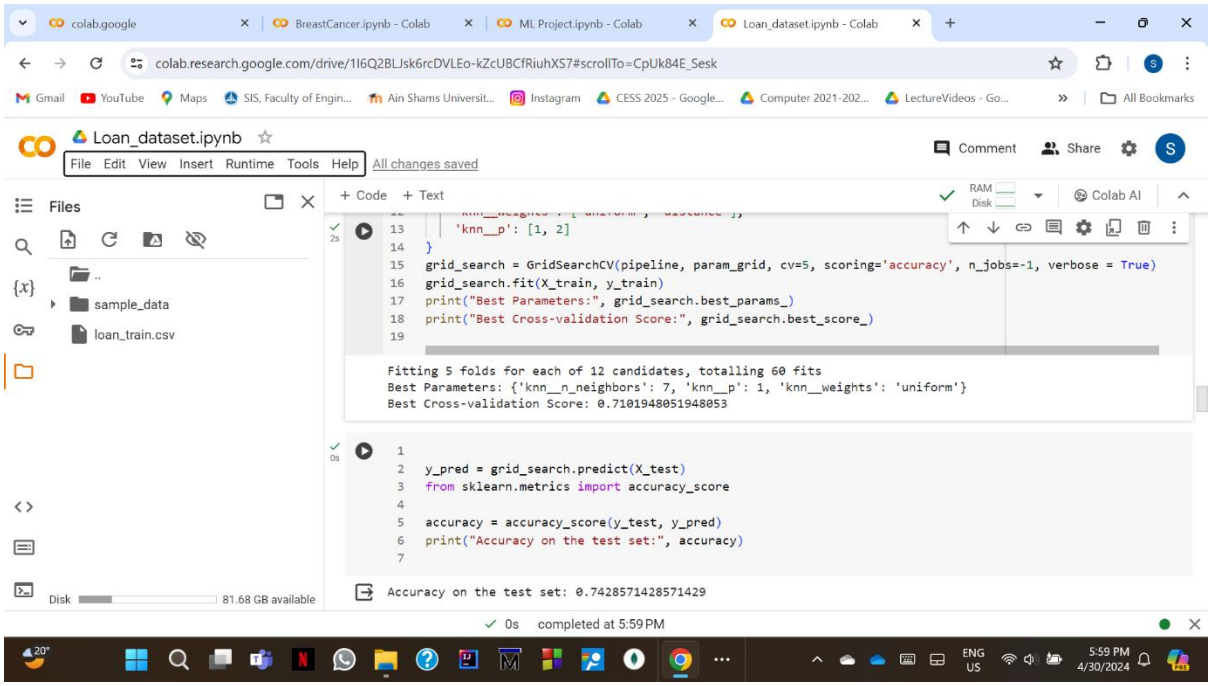
- KNN Model with Parameters (3,5,7,9,11,13) and its accuracy:





- KNN Model with Parameters (3,5,7) and its accuracy:

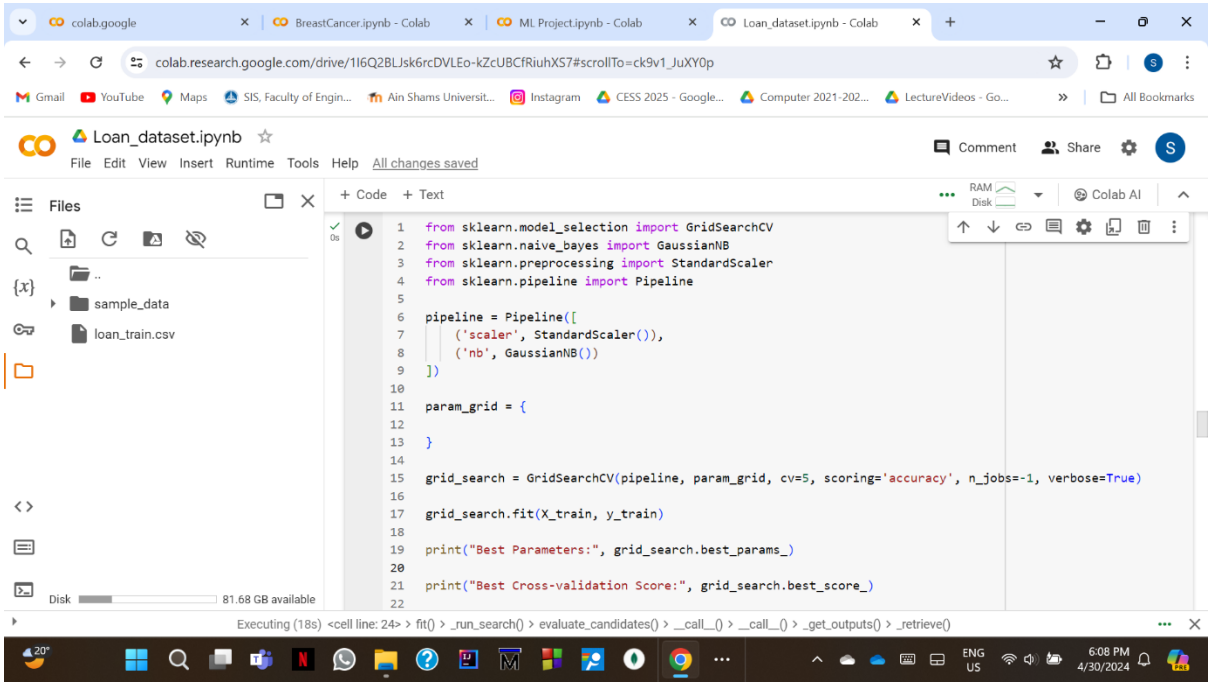


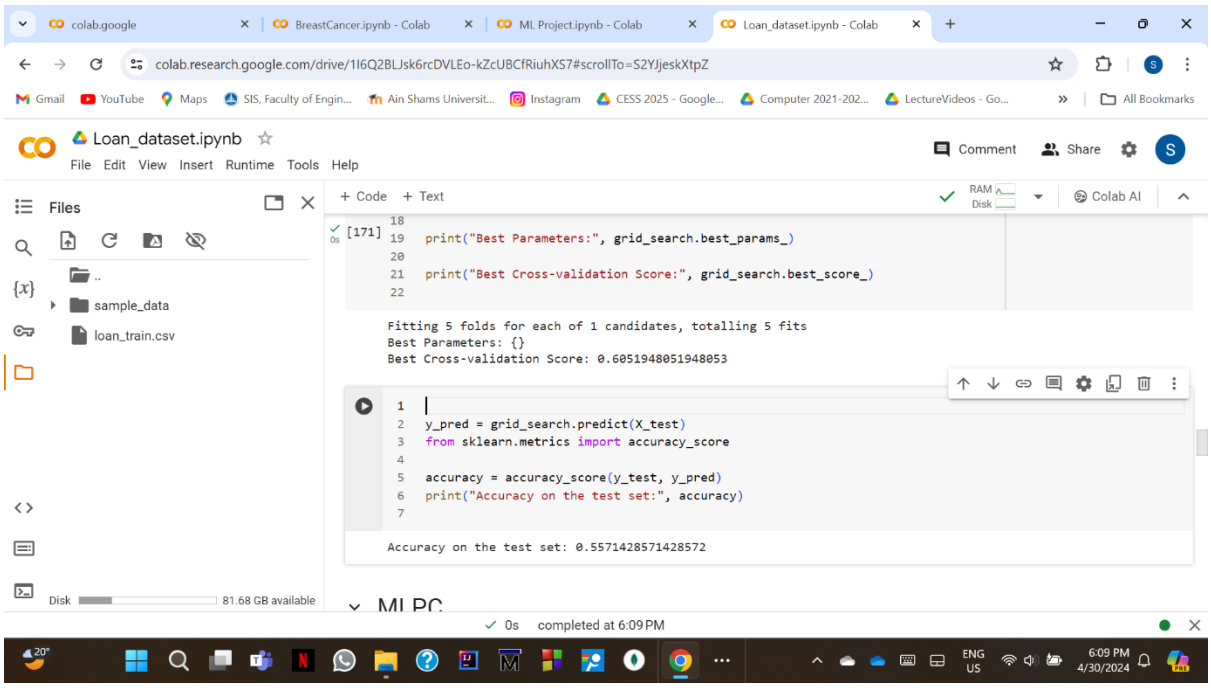


- **The Reason of using the final parameters:**

We notice that in both cases the model has chosen from the neighbour number 7 and to be uniform and using Manhattan which is number 1. Which means that by increasing numbers or decreasing them the best values to be chosen are 7, uniform and 1.

- Naïve Bayes Classifier and its accuracy:





iPhone Purchase:

- **Loading the file:** we used **pd.read_csv()** to read the file as follows:

```
# Load the dataset
df = pd.read_csv('Iphone_purchase.csv')
```

- **Before preprocessing:** we ignored the “id” column from consideration: We used **drop()** function as follows:

```
# Drop the 'User ID' column
df.drop('User ID', axis=1, inplace=True)
```

- **Preprocessing stage:**
  - **Encoding:** Encoding categorical column 'Gender' by **LableEncoder()** as follows:

```
# Encoding categorical column 'Gender'
labelEncoder = LabelEncoder()
labelEncoder.fit(df["Gender"])
df["Gender"] = labelEncoder.transform(df["Gender"])
```

  - **Splitting the data into features (X) and target variable (y) as follows:**

```
# Splitting the data into features (X) and target variable (y)
X = df[['Gender', 'Age', 'EstimatedSalary']]
y = df['Purchased']
```

  - **Splitting the data into training and testing sets:** using **train_test_split()** with test size 20% of the data as follows:

```
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

  - **Feature scaling:** using **StandardScaler()** function on both training and testing inputs as follows:

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

- **Training stage:** we are training two classifiers KNN and Naïve Bayes:
  - **Naïve Bayes:** using **GaussianNB()** as it is the most suitable technique according to the dataset given, we use the scaled training features "x_train_scaled" and training target variables "y_train" by **fit()** function as follows:

```
# Train the Naïve Bayes Classifier
naive_bayes_classifier = GaussianNB()
naive_bayes_classifier.fit(X_train_scaled, y_train)
```

I thought about using Bernoulli instead of Gaussian distribution, but this type works on binarized features only, which contradicts with our training data.

  - **KNN classifier**: I tuned 3 hyperparameters (no. of neighbors, weight of features, metric used). I examined the classifier on ranges of n [1,10] weight [distance, uniform], metric can be (Euclidean or Manhattan). So, we made 3 nested for loop to test all the combinations available as follows:

```
# Train the KNN Classifier with different values of n_neighbors, weights, and metric
for n in range(1, 11):  # trying values from 1 to 10 for n_neighbors hyperparameter
    for weight in ['uniform', 'distance']:  # trying both 'uniform' and 'distance' for weights hyperparameter
        for metric in ['euclidean', 'manhattan']:  # trying only 'euclidean' and 'manhattan' for metric
            knn_classifier = KNeighborsClassifier(n_neighbors=n, weights=weight, metric=metric)
            knn_classifier.fit(X_train_scaled, y_train)
```

- **Testing stage**: we test both classifiers using **predict()** function and measure accuracy using **metrics.accuracy_score()**

  - **Naïve Bayes:** we use the scaled testing data in prediction then we compare testing outputs we have with our predictions to calculate accuracy as follows:

```
# Test the Naïve Bayes Classifier
naive_bayes_predictions = naive_bayes_classifier.predict(X_test_scaled)
naive_bayes_accuracy = accuracy_score(y_test, naive_bayes_predictions)
print("Naïve Bayes Classifier Accuracy:", naive_bayes_accuracy * 100)
print("Naïve Bayes Classifier Predictions on Testing Set:")
print(naive_bayes_predictions)
```

  - **KNN:** we use the same arguments as Naïve Bayes.

```
# Test the KNN Classifier
knn_predictions = knn_classifier.predict(X_test_scaled)
knn_accuracy = accuracy_score(y_test, knn_predictions)
accuracy_scores.append(knn_accuracy)

print(f"KNN Classifier Accuracy (n_neighbors={n}, weights={weight}, metric={metric}):", knn_accuracy * 100)
```

Now, we need to track all the tunings of the hyperparameters to get the combination that gives highest accuracy. Thus, we do some extra steps:

- First, initialize variables to keep track of the best KNN classifier and its hyperparameters as follows:

```
# Initialize variables to keep track of the best KNN classifier and its hyperparameters
best_knn_accuracy = 0
best_knn_predictions = None
best_n_neighbors = None
best_weights = None
best_metric = None
accuracy_scores = []
```

  We set all the best hyperparameters to the minimum to get the highest records.

- Next, we make an if condition that updates the best hyperparameters once better records came as follows:

```
    # Update best KNN classifier and its hyperparameters if current one has higher accuracy
    if knn_accuracy > best_knn_accuracy:
        best_knn_accuracy = knn_accuracy
        best_knn_predictions = knn_predictions
        best_n_neighbors = n
        best_weights = weight
        best_metric = metric
```

- Finally, we print the final best tuning of hyperparameters as follows:

```
print("\n Best KNN Classifier Accuracy:", best_knn_accuracy * 100)
print("Hyperparameters for Best KNN Classifier:")
print("  - n_neighbors:", best_n_neighbors)
print("  - weights:", best_weights)
print("  - metric:", best_metric)
print("Best KNN Classifier Predictions on Testing Set:")
print(best_knn_predictions)
```

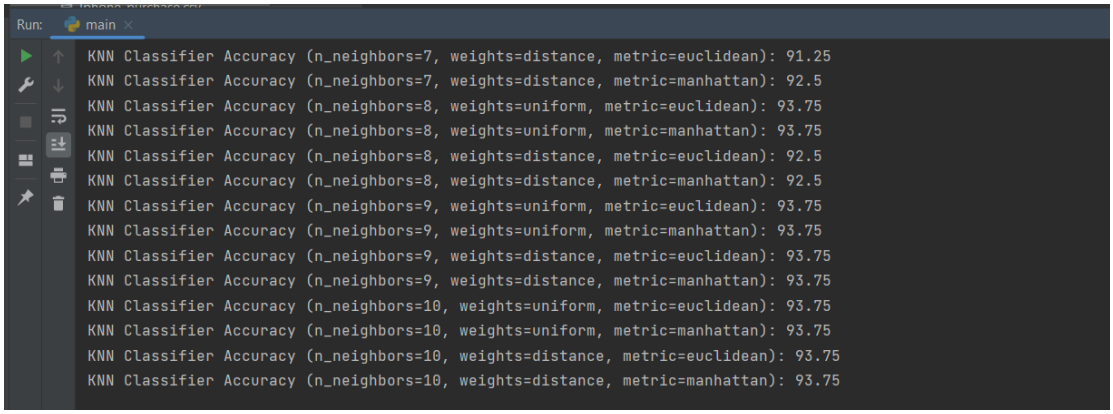
• Screenshots of Output:

- **Naïve Bayes:** No hyperparameters can be tuned in Naïve Bayes classifier except the distribution as using Bernoulli instead of Gaussian but I explained before why such tuning is not good. The output is shown as follows:

```
C:\Users\seifa\pythonProject1\Scripts\python.exe C:\Users\seifa\PycharmProjects\pythonProject1\main.py
Naïve Bayes Classifier Accuracy: 93.75
Naïve Bayes Classifier Predictions on Testing Set:
[1 1 0 1 0 0 1 0 0 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 1 0 0 0 1 0 1 0 0
 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 1 0 0 1 1 1 0 0 1 0 0 0
 0 0 1 1 0 0]
```

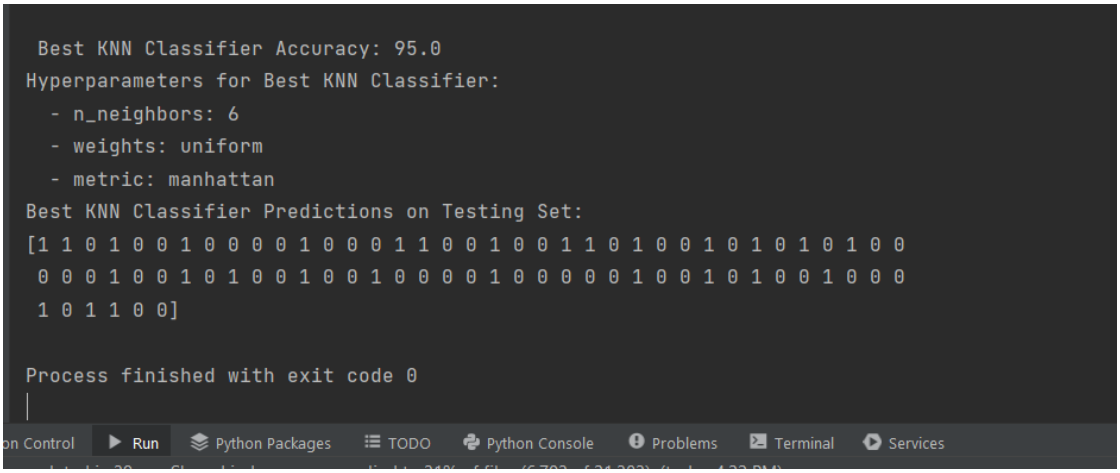
- **KNN:** we have 40 combinations of tuned hyperparameters. (neighbours, weight, metric) shown as follows:

```
Run:  main
KNN Classifier Accuracy (n_neighbors=1, weights=uniform, metric=euclidean): 88.75
KNN Classifier Accuracy (n_neighbors=1, weights=uniform, metric=manhattan): 88.75
KNN Classifier Accuracy (n_neighbors=1, weights=distance, metric=euclidean): 88.75
KNN Classifier Accuracy (n_neighbors=1, weights=distance, metric=manhattan): 88.75
KNN Classifier Accuracy (n_neighbors=2, weights=uniform, metric=euclidean): 83.75
KNN Classifier Accuracy (n_neighbors=2, weights=uniform, metric=manhattan): 85.0
KNN Classifier Accuracy (n_neighbors=2, weights=distance, metric=euclidean): 88.75
KNN Classifier Accuracy (n_neighbors=2, weights=distance, metric=manhattan): 88.75
KNN Classifier Accuracy (n_neighbors=3, weights=uniform, metric=euclidean): 91.25
KNN Classifier Accuracy (n_neighbors=3, weights=uniform, metric=manhattan): 93.75
KNN Classifier Accuracy (n_neighbors=3, weights=distance, metric=euclidean): 88.75
KNN Classifier Accuracy (n_neighbors=3, weights=distance, metric=manhattan): 88.75
KNN Classifier Accuracy (n_neighbors=4, weights=uniform, metric=euclidean): 91.25
KNN Classifier Accuracy (n_neighbors=4, weights=uniform, metric=manhattan): 91.25
KNN Classifier Accuracy (n_neighbors=4, weights=distance, metric=euclidean): 91.25
KNN Classifier Accuracy (n_neighbors=4, weights=distance, metric=manhattan): 91.25
KNN Classifier Accuracy (n_neighbors=5, weights=uniform, metric=euclidean): 92.5
KNN Classifier Accuracy (n_neighbors=5, weights=uniform, metric=manhattan): 92.5
KNN Classifier Accuracy (n_neighbors=5, weights=distance, metric=euclidean): 91.25
KNN Classifier Accuracy (n_neighbors=5, weights=distance, metric=manhattan): 90.0
KNN Classifier Accuracy (n_neighbors=6, weights=uniform, metric=euclidean): 92.5
KNN Classifier Accuracy (n_neighbors=6, weights=uniform, metric=manhattan): 95.0
KNN Classifier Accuracy (n_neighbors=6, weights=distance, metric=euclidean): 92.5
KNN Classifier Accuracy (n_neighbors=6, weights=distance, metric=manhattan): 91.25
KNN Classifier Accuracy (n_neighbors=7, weights=uniform, metric=euclidean): 92.5
KNN Classifier Accuracy (n_neighbors=7, weights=uniform, metric=manhattan): 93.75
KNN Classifier Accuracy (n_neighbors=7, weights=distance, metric=euclidean): 91.25
Version Control  Run  Python Packages  TODO  Python Console  Problems  Terminal  Services
Indexing completed in 39 sec. Shared indexes were applied to 31% of files (6,703 of 21,383). (today 4:22 PM)
```



- **Final Values:**

After we tracked all the combinations of hyperparameters, we were able to output the hyperparameters that recorded the highest accuracies with their predictions as follows:



n_neighbors: **6**
weights: **uniform**
metric: **manhattan**