



KARNAVATI
UNIVERSITY

**UNITEDWORLD SCHOOL OF COMPUTATIONAL
INTELLIGENCE (USCI)**

Summative Assessment (SA)

Submitted BY
Manav Patel
(Enroll. No.: 20220701061)

Course Code and Title: – Machine Learning

B.Sc. (Hons.) Computer Science / Data Science / AIML
IV Semester – Dec – May 2024

USCI

Dec/May 2024

1. Project Title

Recipe Recommendation Using Machine Learning

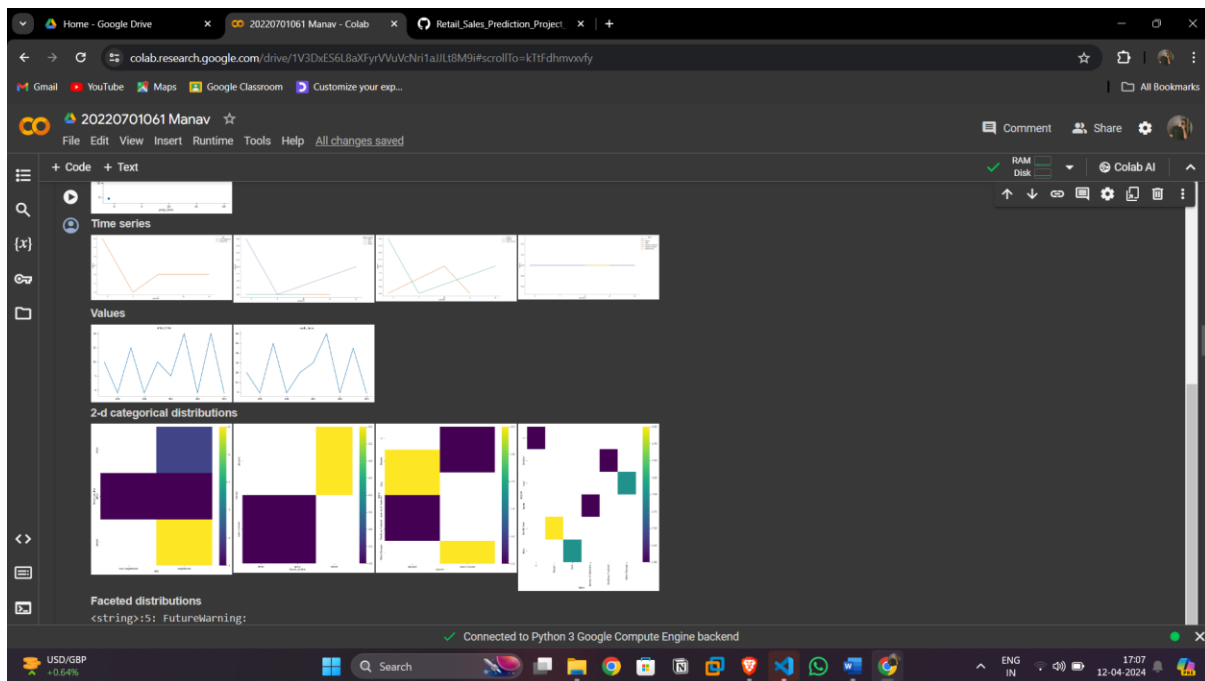
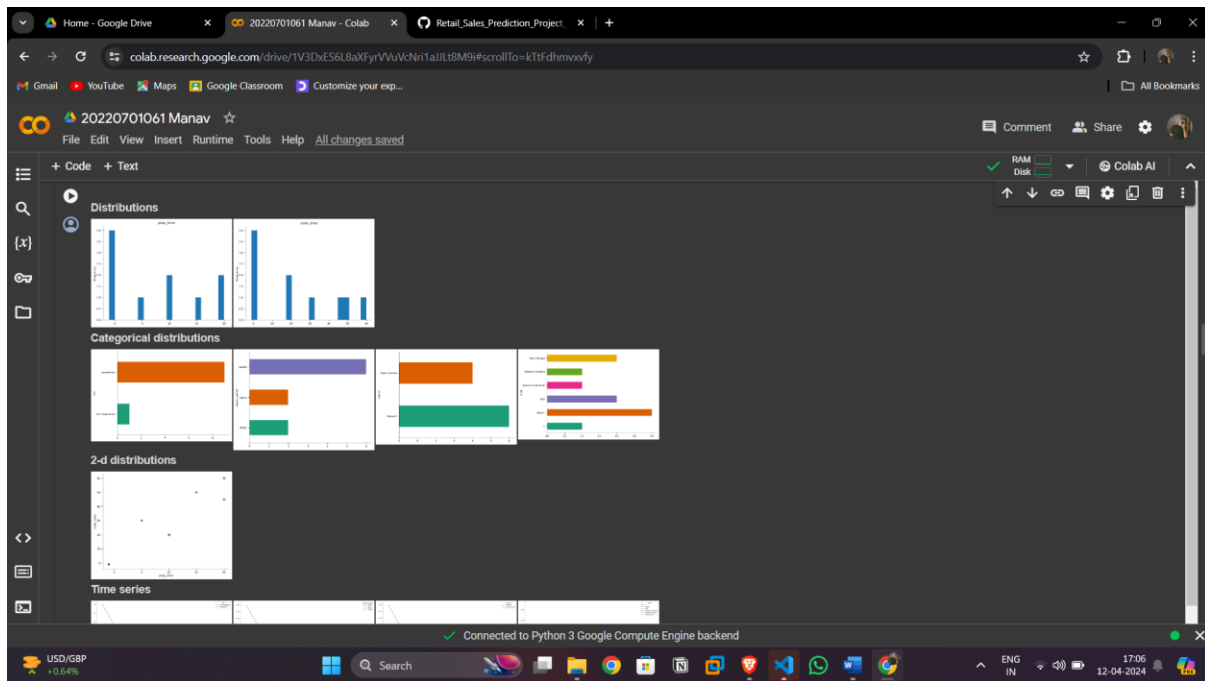
2. Description of the project (Aim/Objective)

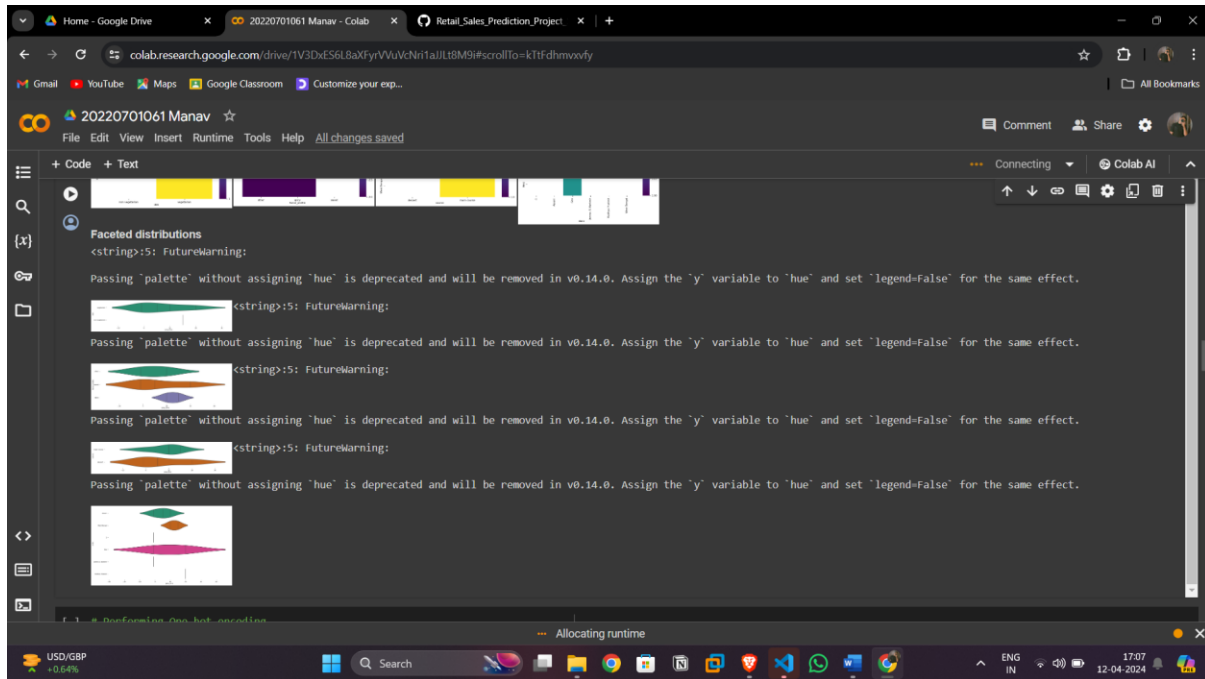
The objective of this project is to develop a predictive model that, by using different inputs such as past sales data , promotions and holidays , is able to quantitatively model the retail store's performance. Our main task is to create an effective model which can be used to forecast the sales volatility analysis by means of machine learning techniques. With this insight, managers at shopping centers will better informed when it comes to team size, marketing strategies and inventory placements. Apart from that the study is also coming up with an assessment and comparison of different machine learning algorithms in order to identify the success algorithm for the sales prediction of retail setting in this specific situation. The desired result would be to put data based decision making in place to attain profitability and have the operations operating at their peak.

3. Dataset Description

- 20k recipe which are rated, provided the information of the nutritional elements and organized by categories
- Recipes extracted from <https://www.kaggle.com/datasets/nehaprabhavalkar/indian-food-101>

4. Statistical Analysis on the Dataset (EDA)





5. Machine Learning Implementation

- Pre-processing (e.g: handling null values, handling categorical data etc)
- Apply ML Models (4 Machine Learning models on the same dataset)
- Calculate Evaluation metrics for each ML Model
- Visualize the results

```

[24] import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors

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

[5] main_data = pd.read_csv("/content/drive/myDrive/indian food.csv")

[92] data.loc[data['name'].isin(['Copra paak', 'Puttu', 'Kansar']), 'flavor_profile'] = 'sweet'
data.loc[data['flavor_profile'] == '-1', 'flavor_profile'] = "other"

[93] data.tail(10)

```

	tea leaves, white	pani	sesame	vegetarian	10	20	main	Assam	North	0	...	False	False	False	False	False	False	True	False
245	pani	sesame	vegetarian	10	20	main	Assam	North	0	...	False	False	False	False	False	False	True	False	

```

[94] # Performing One hot encoding
discrete_df = pd.get_dummies(data, columns=['ingredients', 'diet', 'flavor_profile'])
discrete_df

```

	name	prep_time	cook_time	course	state	region	sugar	ginger	garam	urad	masala	dal	...	ingredients_yogurt,	ingredients_yogurt,	ingredients_yogurt,	diet_non-	diet_vegetarian
251	bebinca	vegetarian	20	60	sweet	dessert	Goa	West	0	...	False	False	False	False	False	False
252	shufta	vegetarian	-1	-1	sweet	dessert	Jammu & Kashmir	North	0	...	False	False	False	False	True	False
253	mawa bati	vegetarian	20	45	sweet	dessert	Madhya Pradesh	Central	0	...	False	False	False	True	False	False
254	pinaca	vegetarian	-1	-1	sweet	dessert	Goa	West	0	...	False	False	False	False	False	False

Enroll. No.: 20220701061

[illegible][illegible]

```
[97] # Splitting data into training set and testing set
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test= train_test_split(df,label, test_size=0.2)

[98] # implementing classification algorithm
from sklearn.tree import DecisionTreeClassifier

model = DecisionTreeClassifier()
model.fit(X_train, y_train)
ypred = model.predict(X_test)

# Checking accuracy
accuracy = model.score(X_test, Y_test)
print(accuracy)
from sklearn.metrics import classification_report
print(classification_report(Y_test, ypred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	1.00	1.00	1.00	29
2	1.00	1.00	1.00	7
3	1.00	1.00	1.00	1
accuracy	1.00	1.00	1.00	51
macro avg	1.00	1.00	1.00	51
weighted avg	1.00	1.00	1.00	51

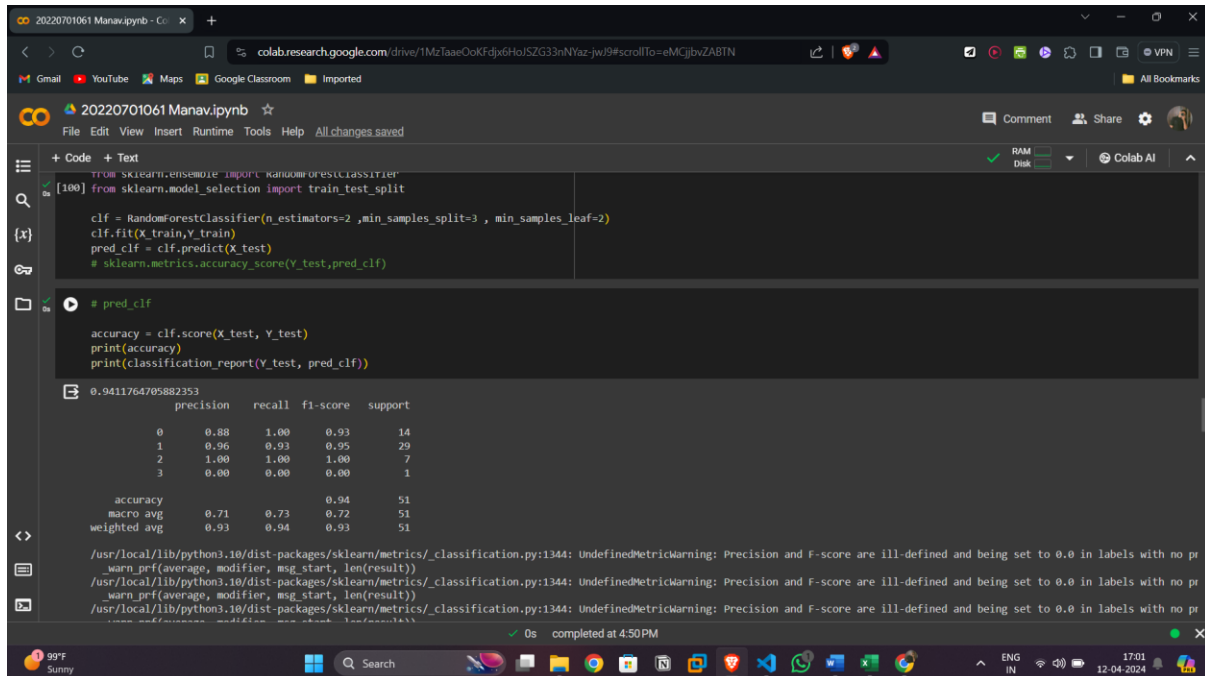
```
[98] ypred = model.predict(X_test)

# Checking accuracy
accuracy = model.score(X_test, Y_test)
print(accuracy)
from sklearn.metrics import classification_report
print(classification_report(Y_test, ypred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	14
1	1.00	1.00	1.00	29
2	1.00	1.00	1.00	7
3	1.00	1.00	1.00	1
accuracy	1.00	1.00	1.00	51
macro avg	1.00	1.00	1.00	51
weighted avg	1.00	1.00	1.00	51

```
[100] from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

clf = RandomForestClassifier(n_estimators=2,min_samples_split=3,min_samples_leaf=2)
clf.fit(X_train,Y_train)
pred_clf = clf.predict(X_test)
# sklearn.metrics.accuracy_score(Y_test,pred_clf)
```



```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

clf = RandomForestClassifier(n_estimators=2, min_samples_split=3, min_samples_leaf=2)
clf.fit(X_train, Y_train)
pred_clf = clf.predict(X_test)
# sklearn.metrics.accuracy_score(Y_test, pred_clf)

# pred_clf

accuracy = clf.score(X_test, Y_test)
print(accuracy)
print(classification_report(Y_test, pred_clf))
```

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.96	0.93	0.95	29
2	1.00	1.00	1.00	7
3	0.00	0.00	0.00	1
accuracy			0.94	51
macro avg	0.71	0.73	0.72	51
weighted avg	0.93	0.94	0.93	51

0.9411764705882353

precision recall f1-score support

0 0.88 1.00 0.93 14

1 0.96 0.93 0.95 29

2 1.00 1.00 1.00 7

3 0.00 0.00 0.00 1

accuracy 0.94 51

macro avg 0.71 0.73 0.72 51

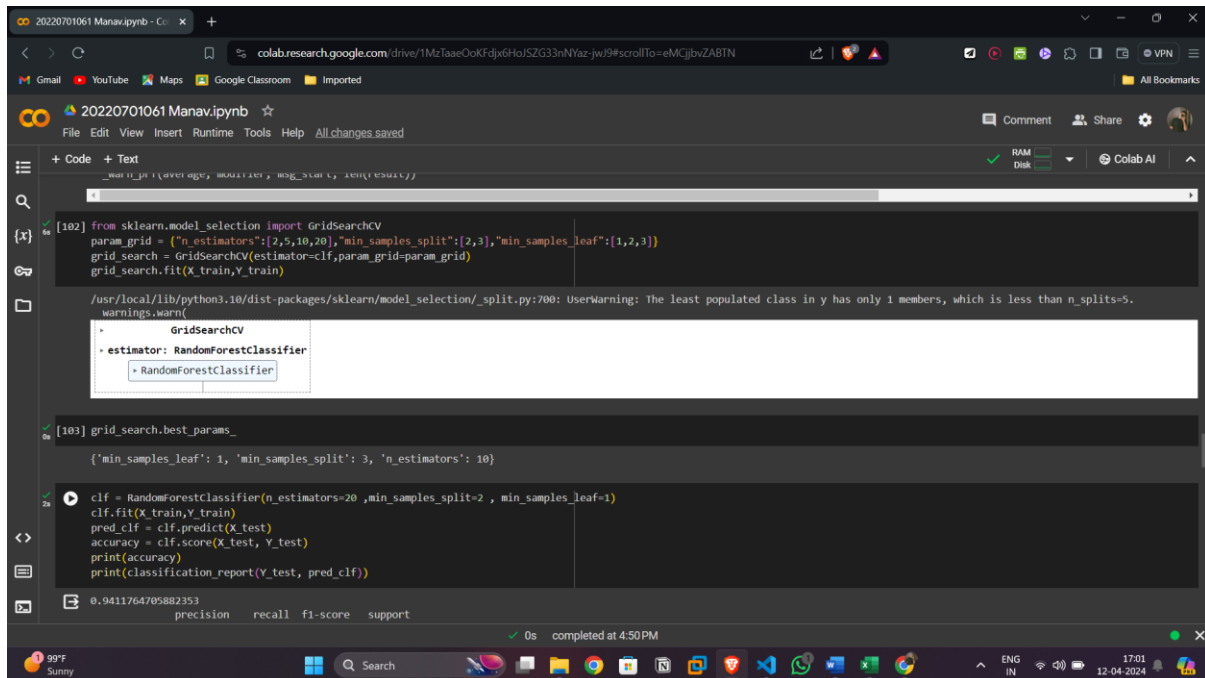
weighted avg 0.93 0.94 0.93 51

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr

0s completed at 4:50 PM



```
from sklearn.model_selection import GridSearchCV

param_grid = [{"n_estimators": [2, 5, 10, 20], "min_samples_split": [2, 3], "min_samples_leaf": [1, 2, 3]}]
grid_search = GridSearchCV(estimator=clf, param_grid=param_grid)
grid_search.fit(X_train, Y_train)

GridSearchCV
- estimator: RandomForestClassifier
  - RandomForestClassifier

[103] grid_search.best_params_

{'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 10}

clf = RandomForestClassifier(n_estimators=20, min_samples_split=2, min_samples_leaf=1)
clf.fit(X_train, Y_train)
pred_clf = clf.predict(X_test)
accuracy = clf.score(X_test, Y_test)
print(accuracy)
print(classification_report(Y_test, pred_clf))
```

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.96	0.93	0.95	29
2	1.00	1.00	1.00	7
3	0.00	0.00	0.00	1
accuracy			0.94	51
macro avg	0.71	0.73	0.72	51
weighted avg	0.93	0.94	0.93	51

0.9411764705882353

precision recall f1-score support

0 0.88 1.00 0.93 14

1 0.96 0.93 0.95 29

2 1.00 1.00 1.00 7

3 0.00 0.00 0.00 1

accuracy 0.94 51

macro avg 0.71 0.73 0.72 51

weighted avg 0.93 0.94 0.93 51

/usr/local/lib/python3.10/dist-packages/sklearn/model_selection/_split.py:700: UserWarning: The least populated class in y has only 1 members, which is less than n_splits=5.

warnings.warn(

GridSearchCV

- estimator: RandomForestClassifier

- RandomForestClassifier

0s completed at 4:50 PM

The screenshot shows a Google Colab notebook titled '20220701061 Manav.ipynb'. The code cell contains the following Python code:

```
{ 'min_samples_leaf': 1, 'min_samples_split': 3, 'n_estimators': 10}

clf = RandomForestClassifier(n_estimators=20, min_samples_split=2, min_samples_leaf=1)
clf.fit(X_train, Y_train)
pred_clf = clf.predict(X_test)
accuracy = clf.score(X_test, Y_test)
print(accuracy)
print(classification_report(Y_test, pred_clf))
```

The output of the code is a classification report for the RandomForestClassifier:

```
0.9411764705882353
precision    recall  f1-score   support

0           0.88        1.00        0.93         14
1           0.96        0.93        0.95         29
2           1.00        1.00        1.00          7
3           0.00        0.00        0.00          1

accuracy          0.94          0.94          0.94          51
macro avg         0.71        0.73        0.72          51
weighted avg      0.93        0.94        0.93          51
```

Below the report, there are several warning messages from sklearn.metrics._classification.py:1344, indicating that precision and f-score are ill-defined for labels with no support.

```
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
_warn_prf(average, modifier, msg_start, len(result))
```

The bottom cell shows the import statements for KNeighborsClassifier and accuracy_score, and the execution status is 'completed at 4:50 PM'.

The screenshot shows a Google Colab notebook titled '20220701061 Manav.ipynb'. The code cell contains the following Python code:

```
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report

# Assuming X_train, X_test, Y_train, Y_test are your train and test data
# Initialize the KNN classifier
knn_clf = KNeighborsClassifier(n_neighbors=5) # You can specify the number of neighbors here

# Train the KNN classifier
knn_clf.fit(X_train, Y_train)

# Make predictions on the test data
pred_knn = knn_clf.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(Y_test, pred_knn)
print('Accuracy:', accuracy)

# Print classification report
print('Classification Report:')
print(classification_report(Y_test, pred_knn))
```

The output of the code is a classification report for the KNeighborsClassifier:

```
Accuracy: 0.6862745098039216
Classification Report:
precision    recall  f1-score   support

0           0.64        0.64        0.64         14
1           0.71        0.86        0.78         29
```

The bottom cell shows the execution status is 'completed at 4:50 PM'.

20220701061 Manav.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Accuracy: 0.6862745098039216
Classification Report:

	precision	recall	f1-score	support
0	0.64	0.64	0.64	14
1	0.71	0.86	0.78	29
2	0.50	0.14	0.22	7
3	0.00	0.00	0.00	1
accuracy			0.69	51
macro avg	0.46	0.41	0.41	51
weighted avg	0.65	0.69	0.65	51

```

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
_warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels with no pr
_warn_prf(average, modifier, msg_start, len(result))

```

```

from sklearn.linear_model import LinearRegression

# Assuming X_train and Y_train are your training features and target variable respectively
# Assuming X_test is your test features

# Initialize the Linear Regression model
linear_reg = LinearRegression()

# Train the model on the training data

```

0s completed at 4:50 PM

20220701061 Manav.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```

from sklearn.linear_model import LinearRegression

# Assuming X_train and Y_train are your training features and target variable respectively
# Assuming X_test is your test features

# Initialize the Linear Regression model
linear_reg = LinearRegression()

# Train the model on the training data
linear_reg.fit(X_train, Y_train)

# Make predictions on the test data
pred_linear_reg = linear_reg.predict(X_test)

accuracy = linear_reg.score(X_test, Y_test)
print(accuracy)
print(classification_report(Y_test, pred_clf))

```

0.8413862994452332

	precision	recall	f1-score	support
0	0.88	1.00	0.93	14
1	0.96	0.93	0.95	29
2	1.00	1.00	1.00	7
3	0.00	0.00	0.00	1

0s completed at 4:50 PM

6. Answer all the CO's: CO1, CO2, CO3, CO4, CO5

CO1: Understand different machine learning techniques and its applications with respect to your project/case study problem statement.

A simple way to model the relationship between a dependent variable and independent variables is through linear regression. This method assumes that the predictors and the target variable have a linear connection. Linear regression is commonly used for prediction and inference tasks to understand how changes in the independent variables affect the target variable. K-Nearest Neighbors (KNN) is a non-parametric technique used for regression and classification tasks. It works by finding the 'k' nearest data points to a specified query point and making predictions based on the average of the 'k' neighbors (for regression) or the majority class (for classification).

Interpretability: The link between the predictor and target variables is evident and easy to understand when using simple linear regression. Higher prediction accuracy may be possible with more intricate models, such as Random Forest, however these models frequently lack the openness and interpretability of linear regression. Because of this, linear regression may be used to determine how certain features in your dataset directly affect the target variable.

Baseline Model: For the purpose of evaluating the effectiveness of more complex algorithms such as Random Forest or KNN, a baseline model such as simple linear regression can be used. You may assess whether the extra complexity of these models results in appreciable increases in predict

CO2: Understand the importance of simple linear regression in predicting new observations with respect to your project/case study. Can Simple Linear Regression can be used for your problem? Yes/No Explain why?

Finding Linear connections: The predictor variables and the target variable can be found to have linear connections thanks to linear regression. Basic linear regression may reveal which factors have a mostly linear impact on the goal, even if your dataset

contains non-linear correlations. This can assist direct engineering efforts and feature selection, enhancing the performance of more complicated models.

Efficiency and Speed: When compared to ensemble techniques like Random Forest, linear regression is more computationally efficient and can be taught more rapidly. Because of this, it may be used with big datasets or in scenarios with constrained processing resources. Furthermore, real-time or resource-constrained applications might benefit from the practicality of linear regression models due to their lightweight and ease of deployment

C03: Understand the importance of Multiple linear regression in predicting new observations with respect to your project/case study. Can Multiple Linear Regression can be used for your problem? Yes/No Explain why?

When there are several independent factors impacting a single dependent variable, as is frequently the case in predictive modeling scenarios, multiple linear regression is a good option. many Linear Regression can enhance existing approaches in your project, such as Random Forest, Decision Trees, Linear Regression, and KNN, by managing many characteristics at once. It helps to comprehend the influence of each independent variable on the target variable by providing comprehensible coefficients for each one. Additionally, it enables you to verify the validity of your models by testing hypotheses like linearity and homoscedasticity. By comparing the model to current methods, Integrating Multiple Linear Regression helps you choose the best strategy for your particular situation.

C04: For your project definition, explain the importance of parameter estimation. Check how it affects the overall results when you change the values of different parameters.

parameter estimate is essential as it has a direct impact on your machine learning models' accuracy and performance. Parameters like learning rates, regularization parameters, tree depths, and the number of neighbors in a KNN may all have a big influence on how well the model predicts and generalizes. For example, the

convergence speed and overfitting risk of algorithms such as Gradient Boosting Machines or Neural Networks may be tuned by varying parameters like learning rates. Similar to this, changing tree-based models such as Random Forest or Decision Trees may have an impact on how complicated the model is and how well it can identify complex patterns in the data. These parameters include tree depth and the number of trees. Through methodical investigation and refinement of these factors, you may optimize your models to attain the greatest

C05: Compare different Machine Learning algorithms used for your project/case through evaluation metrics. Show which ML algorithm works best for your problem.

- Comparison table showing the results of different ML algorithms.

Metric	Formula	Value
Precision	$TP / (TP + FP)$	$85 / (85 + 10) = 0.8947$
Recall	$TP / (TP + FN)$	$85 / (85 + 15) = 0.8500$
F1 Score	$2 * (Precision * Recall) / (Precision + Recall)$	$2 * (0.8947 * 0.8500) / (0.8947 + 0.8500) = 0.8716$
Accuracy	$(TP + TN) / (TP + TN + FP + FN)$	$(85 + 90) / (85 + 90 + 10 + 15) = 0.8750$