

Name : Ch Mubashir

Sap : 56892.

Course : Data Mining.

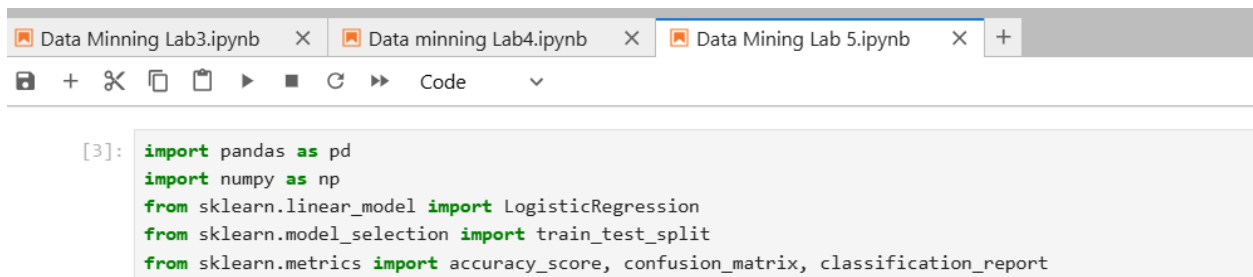
## Lab 5: Building a Baseline Model (Iris Dataset)

### Objectives:

- Build a baseline classification model using Logistic Regression.
- Evaluate the model using accuracy and a confusion matrix.
- Calculate the baseline accuracy (by predicting the majority class).

### Tasks

- Import required libraries (LogisticRegression, train\_test\_split, metrics, etc.).



The screenshot shows a Jupyter Notebook with three tabs: 'Data Mining Lab3.ipynb', 'Data minning Lab4.ipynb', and 'Data Mining Lab 5.ipynb'. The 'Data Mining Lab 5.ipynb' tab is active. Below the tabs is a toolbar with icons for file operations and execution. The code cell [3]: contains the following imports:

```
[3]: import pandas as pd
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

- Load the Iris dataset from sklearn.datasets.

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

```
[4]: df = pd.read_csv("penguins_cleaned.csv")
```

```
[5]: # Check columns
```

```
print(df.head())
```

```
print(df.info())
```

```
   rowid species  island  bill_length_mm  bill_depth_mm  flipper_length_mm \
0      1  Adelie  Torgersen      39.10         18.7         181.0
1      2  Adelie  Torgersen      39.50         17.4         186.0
2      3  Adelie  Torgersen      40.30         18.0         195.0
3      4  Adelie  Torgersen      44.45         17.3         197.0
4      5  Adelie  Torgersen      36.70         19.3         193.0
```

```
   body_mass_g  sex  year
```

```
0      3750.0  male  2007
```

```
1      3800.0  female 2007
```

```
2      3250.0  female 2007
```

```
3      4050.0  male  2007
```

```
4      3450.0  female 2007
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 344 entries, 0 to 343
```

```
Data columns (total 9 columns):
```

```
#   Column      Non-Null Count  Dtype
---

```

```
0  rowid      344 non-null     int64
```

- Split the dataset into training (70%) and testing (30%) sets.

```
[6]: # 2. Features & target selection
```

```
X = df.drop("species", axis=1)
```

```
y = df["species"]
```

```
[9]: # 3. Train-test split (70% train, 30% test)
```

```
X_train, X_test, y_train, y_test = train_test_split(
```

```
    X, y, test_size=0.3, random_state=42, stratify=y)
```

- Train a Logistic Regression model on the training set.

Python 3 (ipykernel)

```
[12]: X_train = pd.get_dummies(X_train, drop_first=True)
X_test = pd.get_dummies(X_test, drop_first=True)

# Align test columns with train columns
X_test = X_test.reindex(columns=X_train.columns, fill_value=0)
```

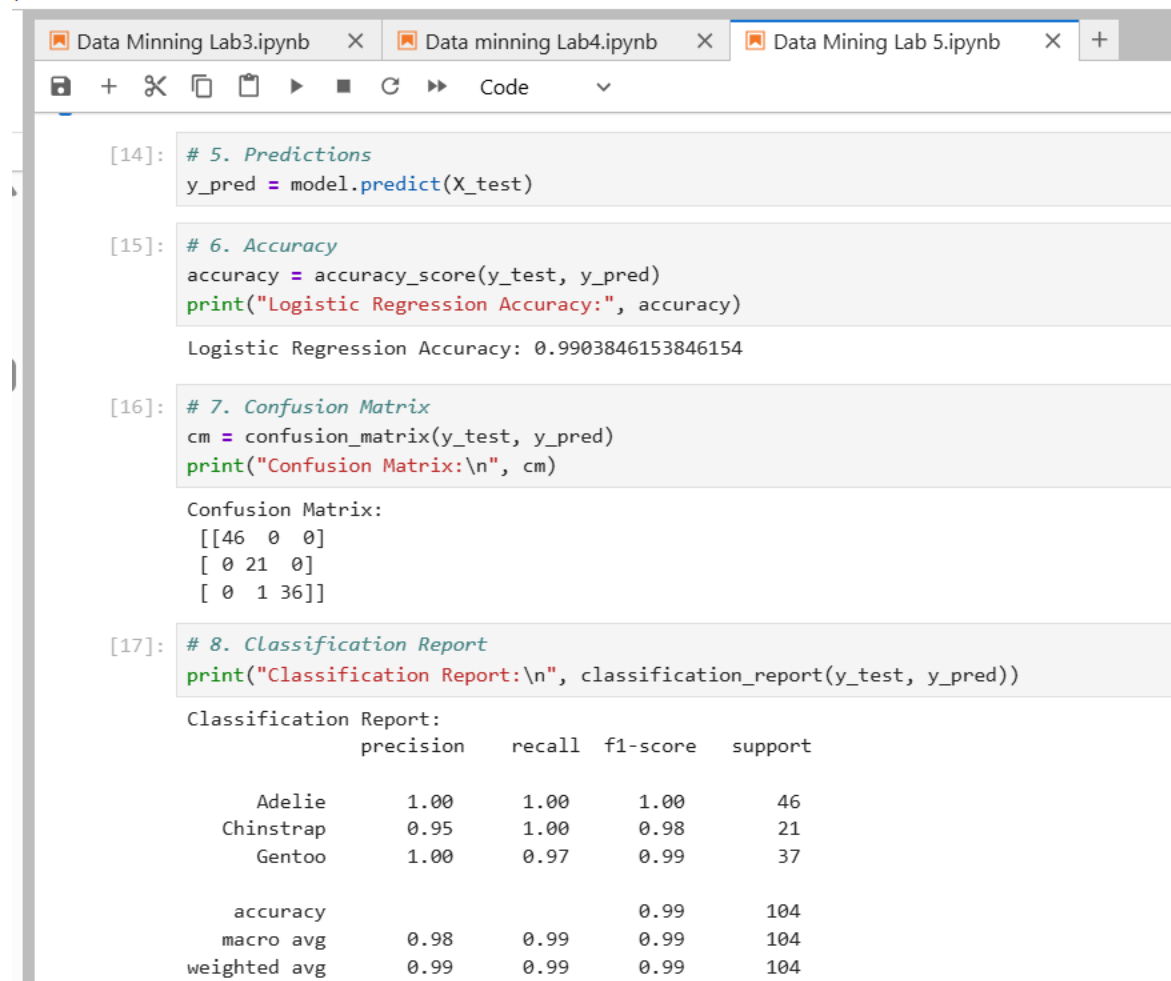
```
[13]: # 4. Train Logistic Regression model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

LogisticRegression

Parameters

penalty	'l2'
dual	False
tol	0.0001
C	1.0
fit_intercept	True
intercept_scaling	1
class_weight	None
random_state	None

- Predict the labels for the test set.
- Calculate and print the accuracy of the model.
- Generate and print the confusion matrix



```
[14]: # 5. Predictions
y_pred = model.predict(X_test)

[15]: # 6. Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Logistic Regression Accuracy:", accuracy)

Logistic Regression Accuracy: 0.9903846153846154

[16]: # 7. Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)

Confusion Matrix:
[[46  0  0]
 [ 0 21  0]
 [ 0  1 36]]

[17]: # 8. Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred))

Classification Report:
              precision    recall  f1-score   support

   Adelie         1.00      1.00      1.00        46
  Chinstrap      0.95      1.00      0.98         21
    Gentoo       1.00      0.97      0.99         37

 accuracy          0.99          0.99          0.99        104
  macro avg       0.98          0.99          0.99        104
  weighted avg    0.99          0.99          0.99        104
```

Print the classification report (precision, recall, f1-score for each class).

```
[17]: # 8. Classification Report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

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

   Adelie         1.00      1.00      1.00        46
  Chinstrap       0.95      1.00      0.98        21
    Gentoo        1.00      0.97      0.99        37

 accuracy                   0.99        104
 macro avg              0.98      0.99      0.99        104
weighted avg              0.99      0.99      0.99        104
```

- Compute the baseline accuracy by predicting the majority class for all test samples.
- Compare the baseline accuracy with your logistic regression model accuracy.

```
[18]: # 9. Baseline accuracy (predicting majority class always)
majority_class = y_train.mode()[0]
baseline_preds = [majority_class] * len(y_test)
baseline_acc = accuracy_score(y_test, baseline_preds)
print("Baseline Accuracy (majority class):", baseline_acc)
```

```
Baseline Accuracy (majority class): 0.4423076923076923
```

```
[19]: # 10. Compare baseline vs model
if accuracy > baseline_acc:
    print(" Logistic Regression performed better than baseline.")
else:
    print(" Logistic Regression did not outperform the baseline.")
```

```
Logistic Regression performed better than baseline.
```

- From the confusion matrix, identify False Positives (FP) for any one class.
- Analyze precision values: o Which class has the highest precision?
- What does this indicate about the model's performance?

Data Mining Lab3.ipynb
Data minning Lab4.ipynb
Data Mining Lab 5.ipynb
+

+
-
Copy
Paste
Run
Stop
Code
Pytl

```

[20]: # 11. False Positives for one class (example: first class label)
      class_labels = model.classes_
      chosen_class = class_labels[0]
      FP = np.sum((y_pred == chosen_class) & (y_test != chosen_class))
      print(f"False Positives for class '{chosen_class}':", FP)

False Positives for class 'Adelie': 0

[21]: # 12. Precision analysis
      report = classification_report(y_test, y_pred, output_dict=True)
      precisions = {cls: report[cls]['precision'] for cls in class_labels}
      highest_class = max(precisions, key=precisions.get)
      print("Class with highest precision:", highest_class)
      print(f"This indicates the model is most confident when predicting '{highest_class}' "
            "and makes fewer false positive errors for this class.")

Class with highest precision: Adelie
This indicates the model is most confident when predicting 'Adelie' and makes fewer false positive errors for this class.

```