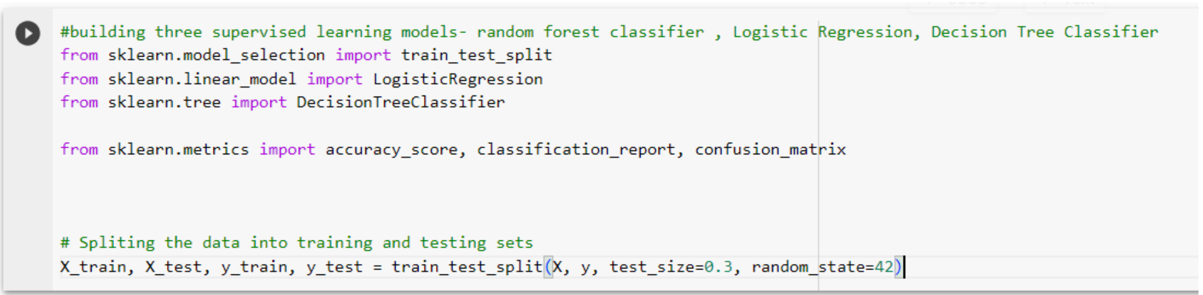
DATA-A-THON:

TEAM newSENSE (#3)

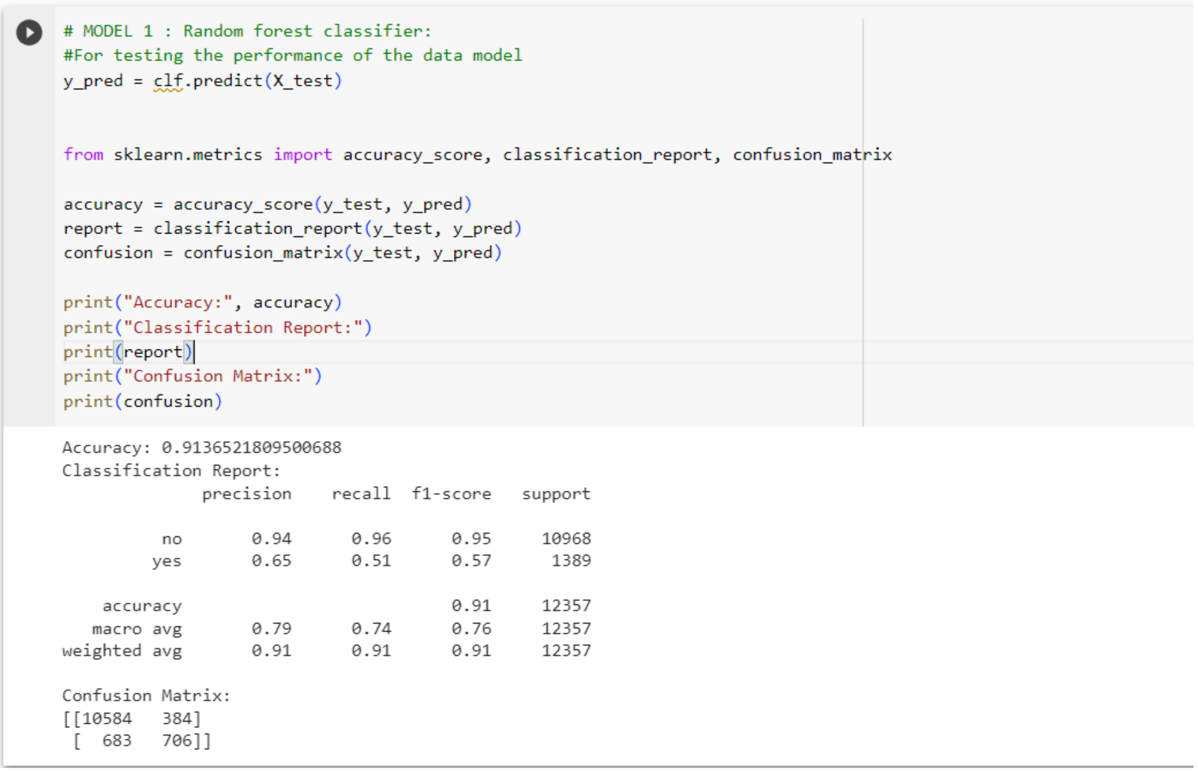
S.Mithilesh Gopalakrishnan
Gayathrisri . G

SLOT-2

3. Build the Supervised Learning Models (Any three).

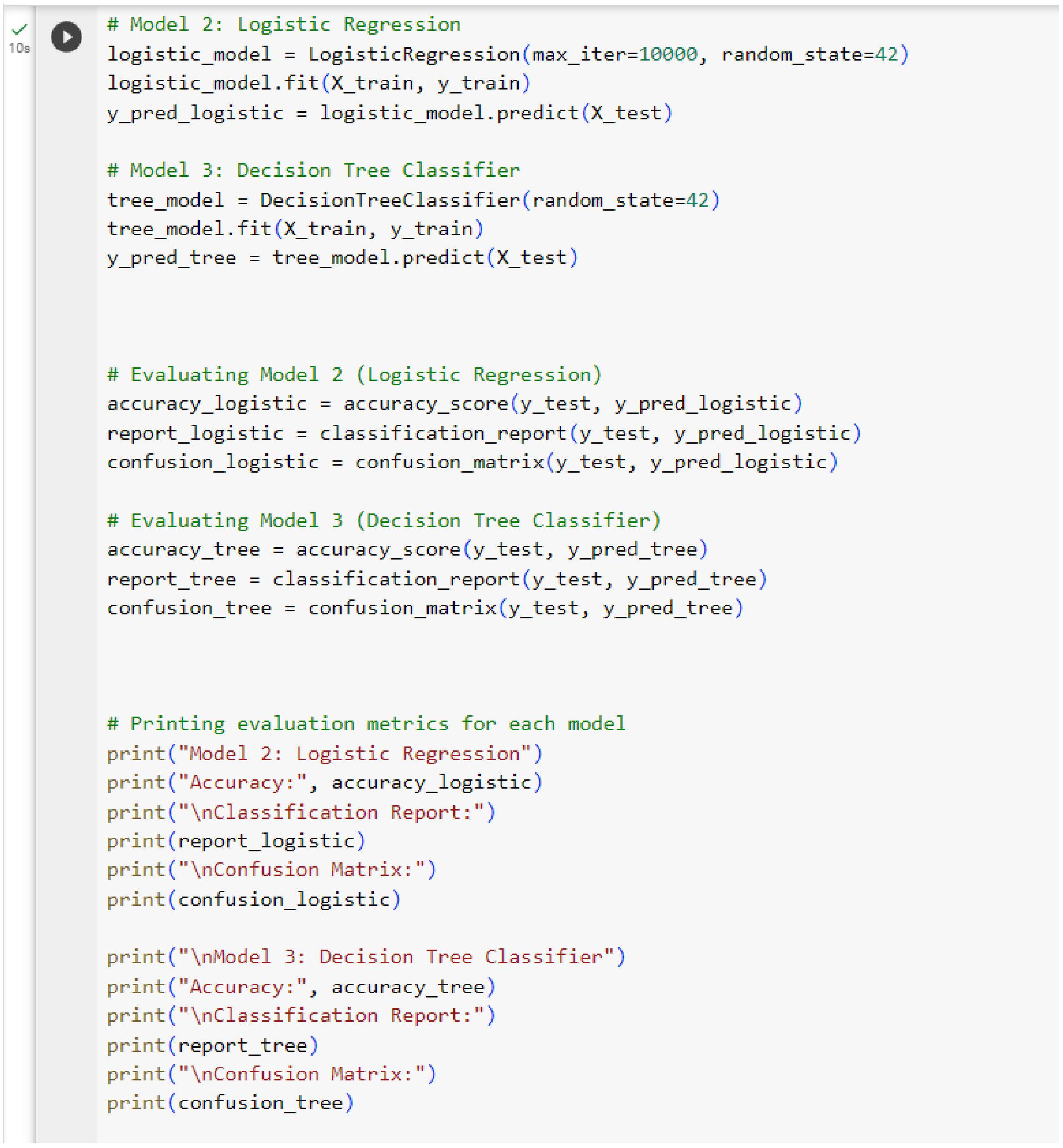


```
#building three supervised learning models- random forest classifier , Logistic Regression, Decision Tree Classifier
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy_score, classification_report, confusion_matrix



# Spliting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

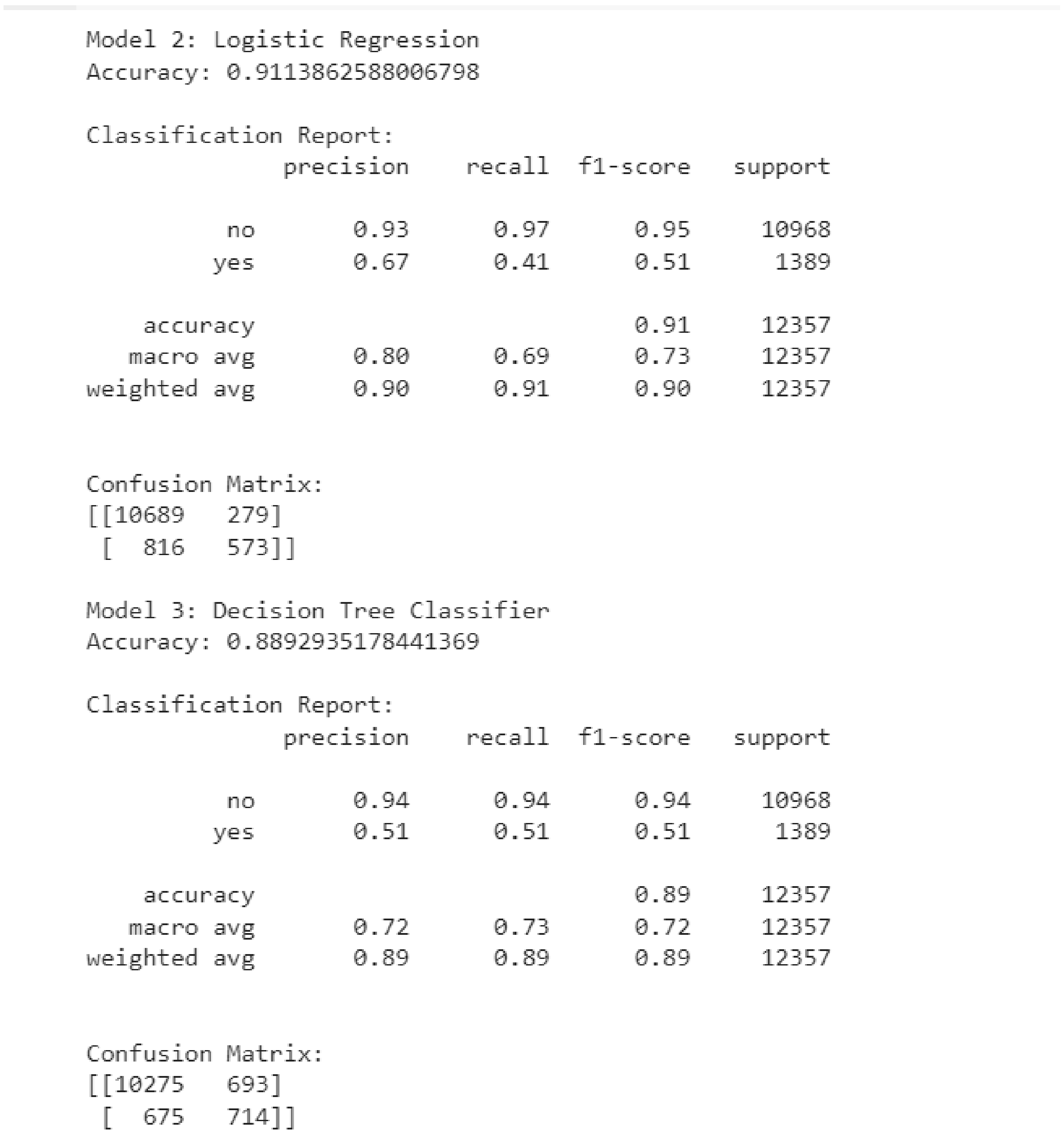


```
# MODEL 1 : Random forest classifier:
#For testing the performance of the data model
y_pred = clf.predict(X_test)


from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

print("Accuracy:", accuracy)
print("Classification Report:")
print(report)
print("Confusion Matrix:")
print(confusion)
```

```
Accuracy: 0.9136521809500688
Classification Report:
              precision    recall  f1-score   support

          no       0.94      0.96      0.95     10968
         yes       0.65      0.51      0.57      1389

    accuracy                           0.91     12357
   macro avg       0.79      0.74      0.76     12357
weighted avg       0.91      0.91      0.91     12357

Confusion Matrix:
[[10584   384]
 [  683   706]]
```



```python
# Model 2: Logistic Regression
logistic_model = LogisticRegression(max_iter=10000, random_state=42)
logistic_model.fit(X_train, y_train)
y_pred_logistic = logistic_model.predict(X_test)

# Model 3: Decision Tree Classifier
tree_model = DecisionTreeClassifier(random_state=42)
tree_model.fit(X_train, y_train)
y_pred_tree = tree_model.predict(X_test)



# Evaluating Model 2 (Logistic Regression)
accuracy_logistic = accuracy_score(y_test, y_pred_logistic)
report_logistic = classification_report(y_test, y_pred_logistic)
confusion_logistic = confusion_matrix(y_test, y_pred_logistic)

# Evaluating Model 3 (Decision Tree Classifier)
accuracy_tree = accuracy_score(y_test, y_pred_tree)
report_tree = classification_report(y_test, y_pred_tree)
confusion_tree = confusion_matrix(y_test, y_pred_tree)



# Printing evaluation metrics for each model
print("Model 2: Logistic Regression")
print("Accuracy:", accuracy_logistic)
print("\nClassification Report:")
print(report_logistic)
print("\nConfusion Matrix:")
print(confusion_logistic)

print("\nModel 3: Decision Tree Classifier")
print("Accuracy:", accuracy_tree)
print("\nClassification Report:")
print(report_tree)
print("\nConfusion Matrix:")
print(confusion_tree)
```



```
Model 2: Logistic Regression
Accuracy: 0.9113862588006798

Classification Report:
              precision    recall  f1-score   support

          no       0.93      0.97      0.95     10968
         yes       0.67      0.41      0.51      1389

    accuracy                           0.91     12357
   macro avg       0.80      0.69      0.73     12357
weighted avg       0.90      0.91      0.90     12357


Confusion Matrix:
[[10689   279]
 [  816   573]]

Model 3: Decision Tree Classifier
Accuracy: 0.8892935178441369

Classification Report:
              precision    recall  f1-score   support

          no       0.94      0.94      0.94     10968
         yes       0.51      0.51      0.51      1389

    accuracy                           0.89     12357
   macro avg       0.72      0.73      0.72     12357
weighted avg       0.89      0.89      0.89     12357


Confusion Matrix:
[[10275   693]
 [  675   714]]
```

4. Choice of Evaluation Metric (Which metric you use and why?)

Support:

Support represents the number of occurrences of each class in your dataset. In a binary classification problem, it's the number of instances in the positive (or "yes") class and the number of instances in the negative (or "no") class.

Use: Support is useful for understanding class distribution. It helps identify class imbalances and assess whether one class is significantly smaller than the other. Imbalanced classes can affect model performance, and support provides insights into this issue.

Confusion Matrix:

Definition: A confusion matrix is a table that summarizes the performance of a classification algorithm. It shows the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN).

Use:

Accuracy: It provides an overall measure of the model's correctness. Accuracy is calculated as (TP + TN) / (TP + TN + FP + FN), where TP and TN are the correctly predicted instances of the positive and negative classes, respectively.

Precision: Precision is the ratio of true positives to the total predicted positives, calculated as TP / (TP + FP). It measures the model's ability to correctly identify positive instances and minimize false positives.

Recall (Sensitivity): Recall is the ratio of true positives to the total actual positives, calculated as TP / (TP + FN). It measures the model's ability to identify all positive instances and minimize false negatives.

F1-Score: F1-score is the harmonic mean of precision and recall, calculated as 2 * (Precision * Recall) / (Precision + Recall). It balances precision and recall, making it useful when class distribution is imbalanced.

Choice of Metric:

Accuracy is a common metric for balanced datasets but can be misleading when classes are imbalanced.

Precision and **recall** are important when the cost of false positives and false negatives differs significantly. For example, in medical diagnosis, recall might be more critical than precision.

F1-score is suitable when you want to balance precision and recall.

Support and the confusion matrix help provide context and insights into the model's performance and class distribution.

5. Overfitting avoidance mechanism:

We chose feature selection to help improve model interpretability and reduce the risk of overfitting by focusing on the most informative features.

Common feature selection methods include filter methods (e.g., mutual information, chi-squared