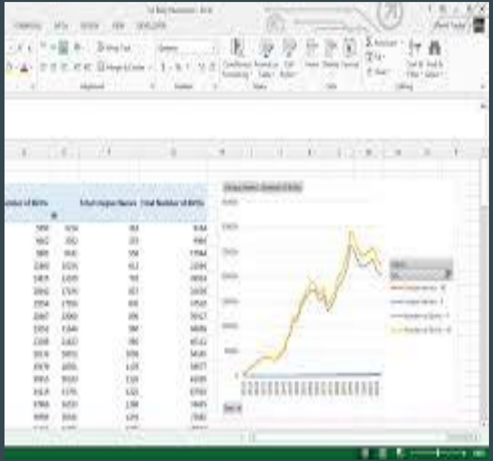
jupyterLab



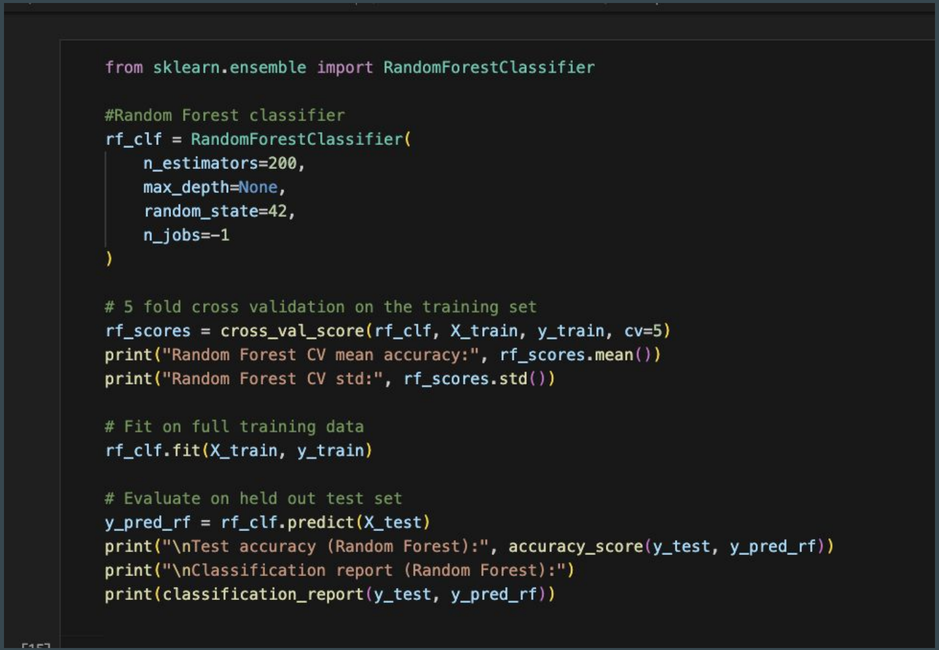
Signal Savants

By Shaheer Siddiqi

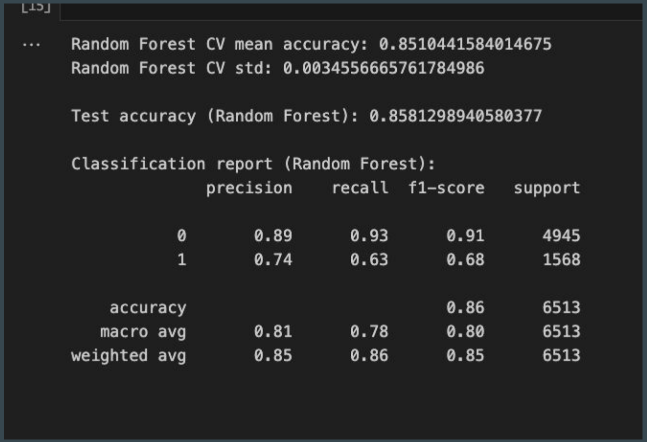
Dataset #1

Tabular - Classification

Adult Census Income

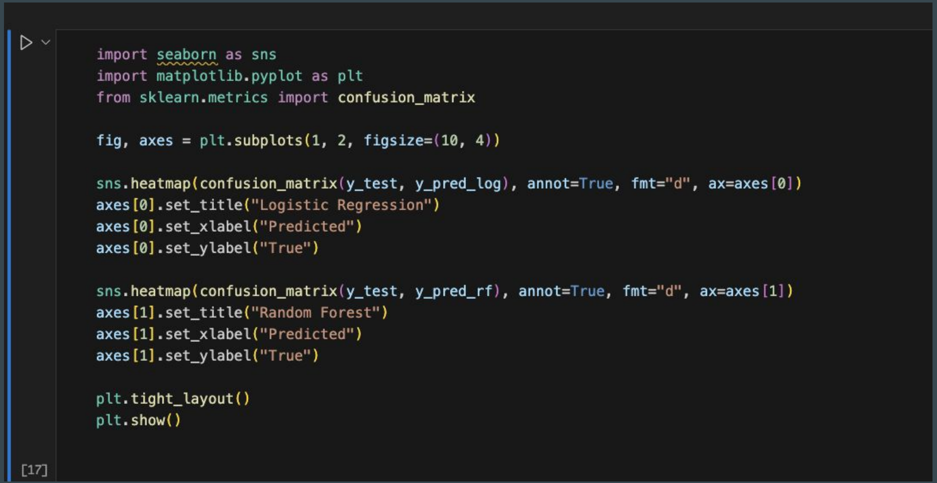


```
from sklearn.ensemble import RandomForestClassifier

#Random Forest classifier
rf_clf = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    random_state=42,
    n_jobs=-1
)

# 5 fold cross validation on the training set
rf_scores = cross_val_score(rf_clf, X_train, y_train, cv=5)
print("Random Forest CV mean accuracy:", rf_scores.mean())
print("Random Forest CV std:", rf_scores.std())

# Fit on full training data
rf_clf.fit(X_train, y_train)

# Evaluate on held out test set
y_pred_rf = rf_clf.predict(X_test)
print("\nTest accuracy (Random Forest):", accuracy_score(y_test, y_pred_rf))
print("\nClassification report (Random Forest):")
print(classification_report(y_test, y_pred_rf))
```



```
Random Forest CV mean accuracy: 0.8510441584014675
Random Forest CV std: 0.0034556665761784986

Test accuracy (Random Forest): 0.8581298940580377

Classification report (Random Forest):
              precision    recall  f1-score   support

           0       0.89      0.93      0.91      4945
           1       0.74      0.63      0.68      1568

    accuracy                           0.86      6513
   macro avg       0.81      0.78      0.80      6513
weighted avg       0.85      0.86      0.85      6513
```

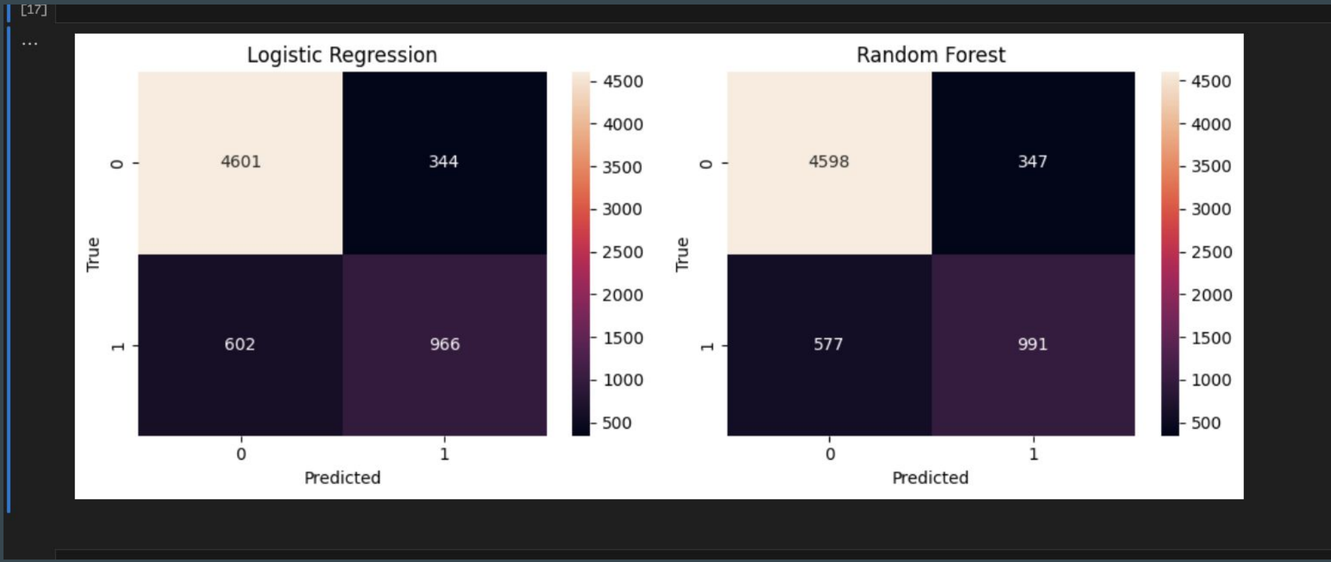
Two classification models were trained on the CYMBAL tabular dataset. Random Forest captures complex, non-linear relationships using multiple decision trees, while Logistic Regression serves as a baseline model to identify linear relationships between features and the target variable.



```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

fig, axes = plt.subplots(1, 2, figsize=(10, 4))

sns.heatmap(confusion_matrix(y_test, y_pred_log), annot=True, fmt="d", ax=axes[0])
axes[0].set_title("Logistic Regression")
axes[0].set_xlabel("Predicted")
axes[0].set_ylabel("True")

sns.heatmap(confusion_matrix(y_test, y_pred_rf), annot=True, fmt="d", ax=axes[1])
axes[1].set_title("Random Forest")
axes[1].set_xlabel("Predicted")
axes[1].set_ylabel("True")

plt.tight_layout()
plt.show()
```

[17]

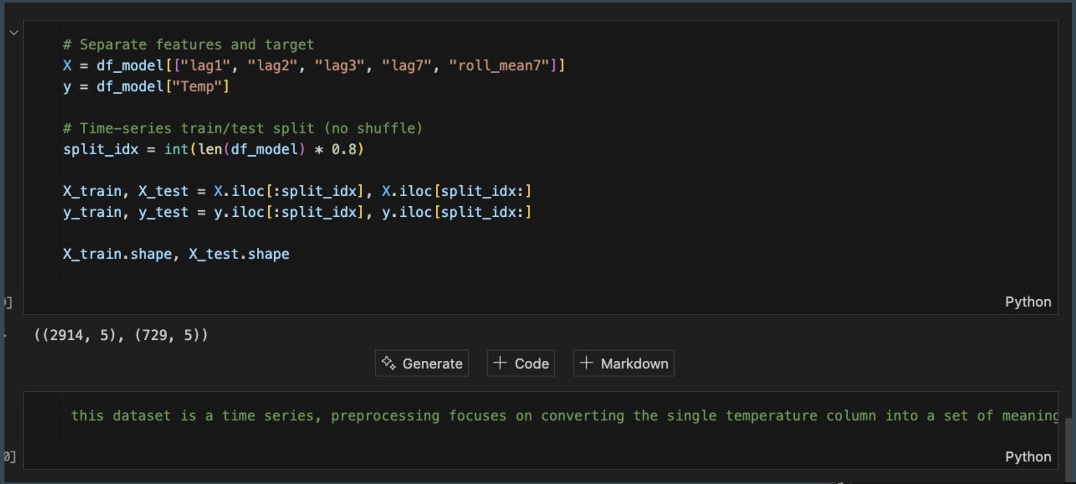
Model performance was evaluated using a confusion matrix generated with the sklearn metrics package. The confusion matrix visualizes correct predictions and highlights misclassifications across different classes.



Dataset #2

Time Series - Regression

Air Quality

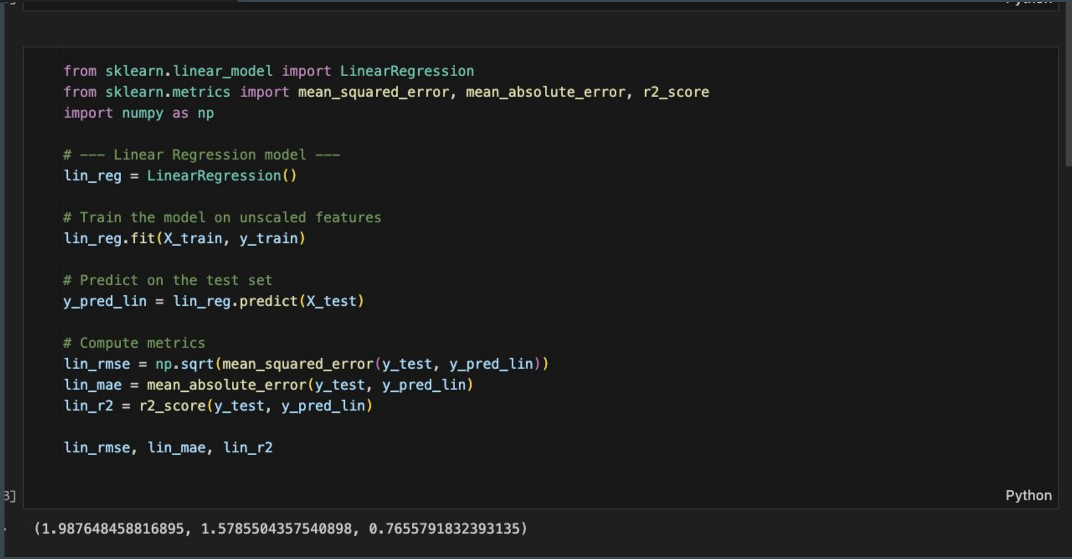


```
# Separate features and target
X = df_model[["lag1", "lag2", "lag3", "lag7", "roll_mean7"]]
y = df_model["Temp"]

# Time-series train/test split (no shuffle)
split_idx = int(len(df_model) * 0.8)

X_train, X_test = X.iloc[:split_idx], X.iloc[split_idx:]
y_train, y_test = y.iloc[:split_idx], y.iloc[split_idx:]

X_train.shape, X_test.shape
```

```
((2914, 5), (729, 5))
```

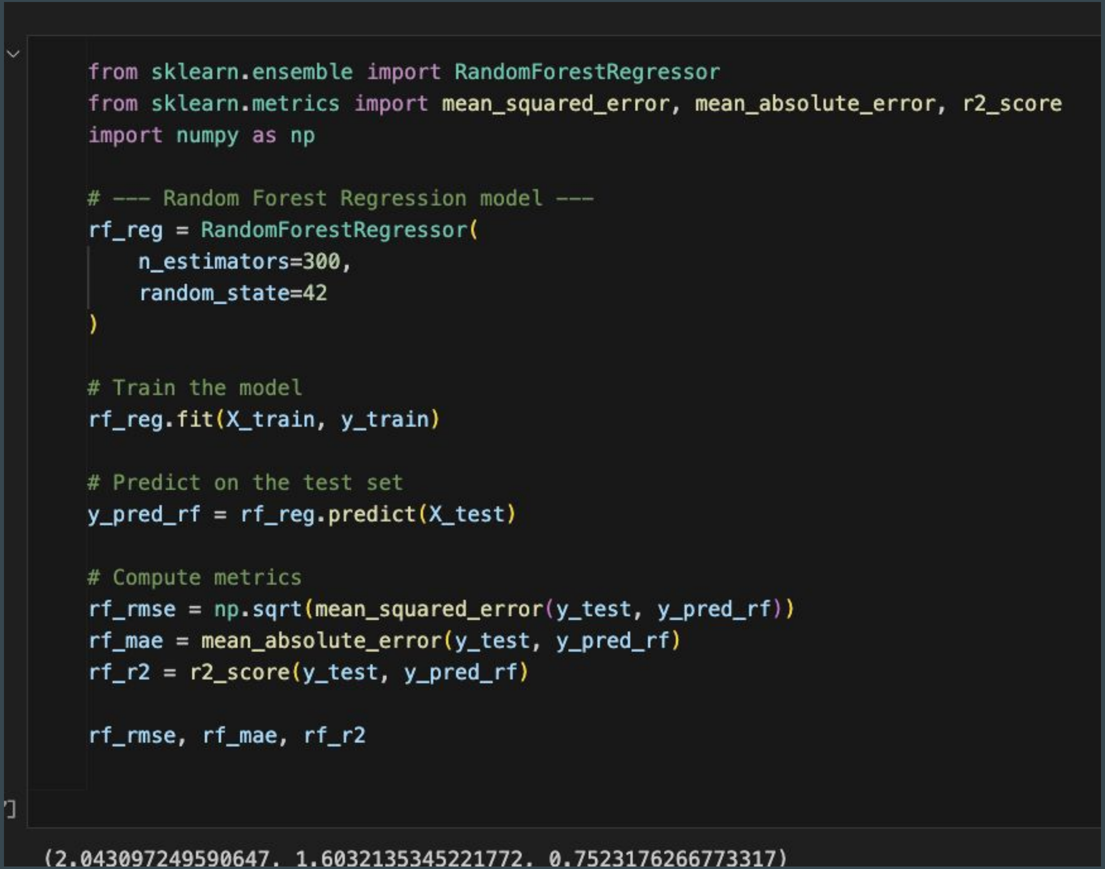
```
this dataset is a time series, preprocessing focuses on converting the single temperature column into a set of meaning
```

This dataset consists of unstructured text data from CYMBAL product descriptions. Text data requires preprocessing and feature extraction before machine learning models can be applied.



```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# --- Linear Regression model ---
lin_reg = LinearRegression()

# Train the model on unscaled features
lin_reg.fit(X_train, y_train)

# Predict on the test set
y_pred_lin = lin_reg.predict(X_test)

# Compute metrics
lin_rmse = np.sqrt(mean_squared_error(y_test, y_pred_lin))
lin_mae = mean_absolute_error(y_test, y_pred_lin)
lin_r2 = r2_score(y_test, y_pred_lin)

lin_rmse, lin_mae, lin_r2
```

```
(1.987648458816895, 1.5785504357540898, 0.7655791832393135)
```



```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import numpy as np

# --- Random Forest Regression model ---
rf_reg = RandomForestRegressor(
    n_estimators=300,
    random_state=42
)

# Train the model
rf_reg.fit(X_train, y_train)

# Predict on the test set
y_pred_rf = rf_reg.predict(X_test)

# Compute metrics
rf_rmse = np.sqrt(mean_squared_error(y_test, y_pred_rf))
rf_mae = mean_absolute_error(y_test, y_pred_rf)
rf_r2 = r2_score(y_test, y_pred_rf)

rf_rmse, rf_mae, rf_r2
```

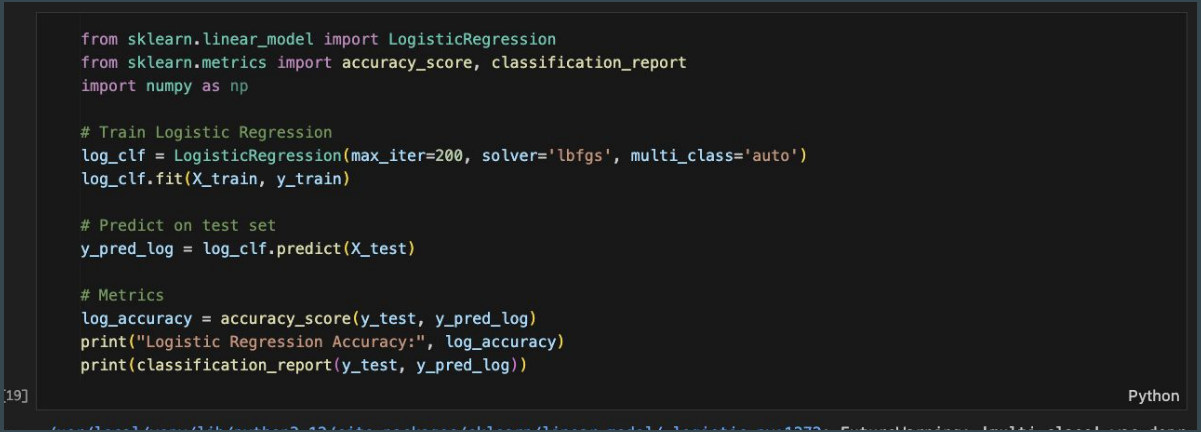
```
(2.043097249590647, 1.6032135345221772, 0.7523176266773317)
```

Text preprocessing was performed using the Natural Language Toolkit. Steps included lowercasing text, removing stop words, tokenizing sentences, and eliminating non-alphabetic characters to reduce noise and improve consistency.

Dataset #3

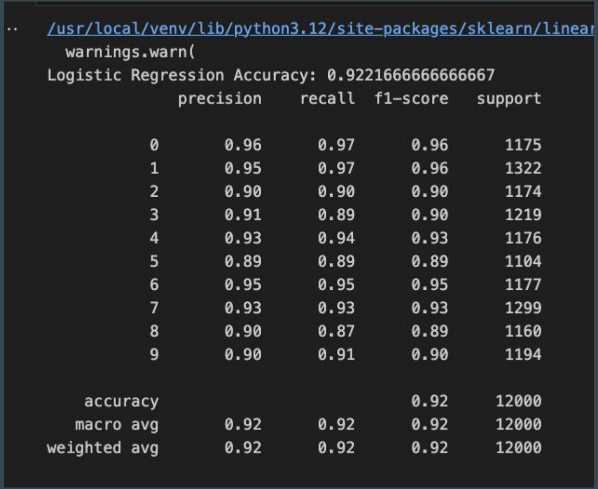
Imaging - Classification

Handwritten Digit



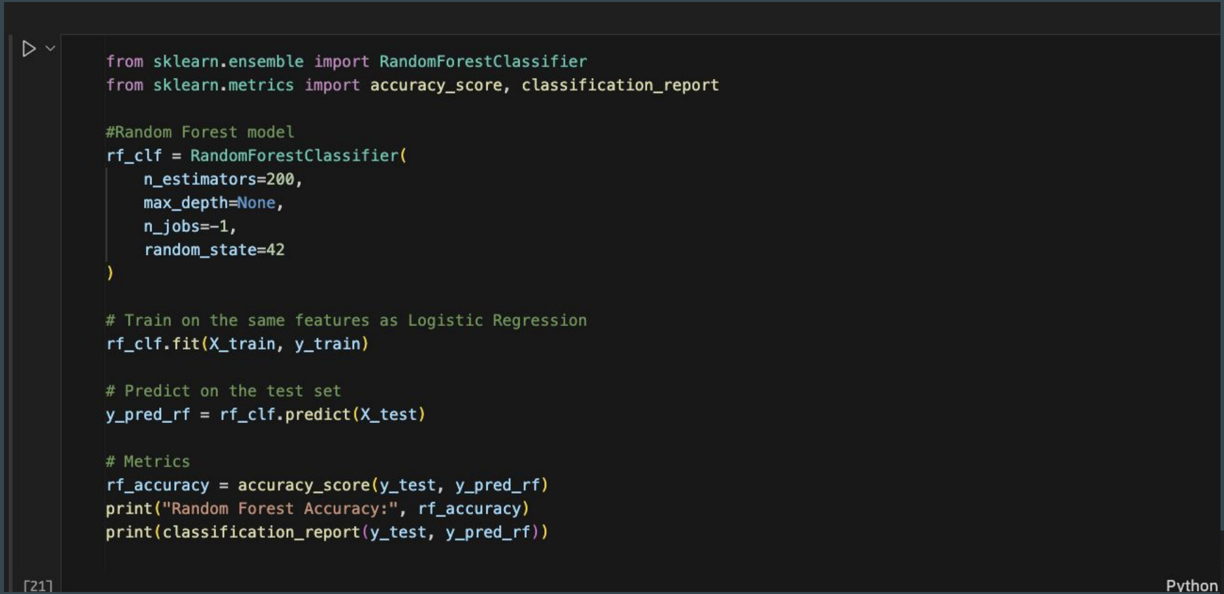
```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import numpy as np

# Train Logistic Regression
log_clf = LogisticRegression(max_iter=200, solver='lbfgs', multi_class='auto')
log_clf.fit(X_train, y_train)

# Predict on test set
y_pred_log = log_clf.predict(X_test)

# Metrics
log_accuracy = accuracy_score(y_test, y_pred_log)
print("Logistic Regression Accuracy:", log_accuracy)
print(classification_report(y_test, y_pred_log))
```

[19] Python

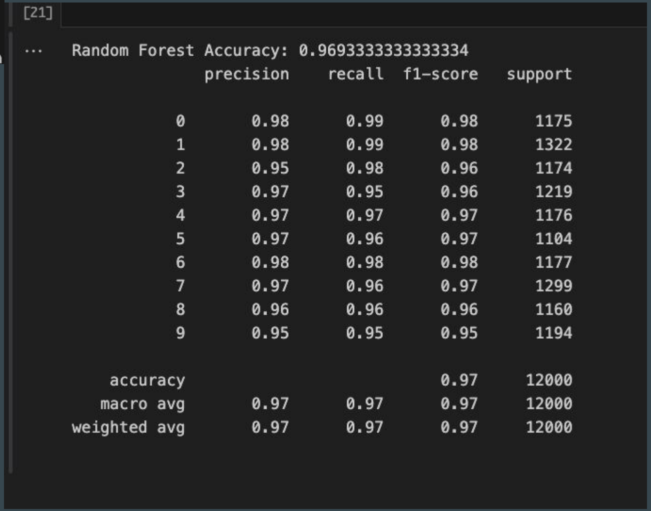


```
/usr/local/venv/lib/python3.12/site-packages/sklearn/linear
  warnings.warn(
Logistic Regression Accuracy: 0.9221666666666667
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      1175
           1       0.95      0.97      0.96      1322
           2       0.90      0.90      0.90      1174
           3       0.91      0.89      0.90      1219
           4       0.93      0.94      0.93      1176
           5       0.89      0.89      0.89      1104
           6       0.95      0.95      0.95      1177
           7       0.93      0.93      0.93      1299
           8       0.90      0.87      0.89      1160
           9       0.90      0.91      0.90      1194

    accuracy                           0.92     12000
   macro avg       0.92      0.92      0.92     12000
weighted avg       0.92      0.92      0.92     12000
```

KMeans and KMedians clustering models from the sklearn cluster module were applied to group similar product descriptions. These methods cluster data based on similarity using mean-based and median-based approaches.



```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

#Random Forest model
rf_clf = RandomForestClassifier(
    n_estimators=200,
    max_depth=None,
    n_jobs=-1,
    random_state=42
)

# Train on the same features as Logistic Regression
rf_clf.fit(X_train, y_train)

# Predict on the test set
y_pred_rf = rf_clf.predict(X_test)

# Metrics
rf_accuracy = accuracy_score(y_test, y_pred_rf)
print("Random Forest Accuracy:", rf_accuracy)
print(classification_report(y_test, y_pred_rf))
```



```
Random Forest Accuracy: 0.9693333333333334
              precision    recall  f1-score   support

           0       0.98      0.99      0.98      1175
           1       0.98      0.99      0.98      1322
           2       0.95      0.98      0.96      1174
           3       0.97      0.95      0.96      1219
           4       0.97      0.97      0.97      1176
           5       0.97      0.96      0.97      1104
           6       0.98      0.98      0.98      1177
           7       0.97      0.96      0.97      1299
           8       0.96      0.96      0.96      1160
           9       0.95      0.95      0.95      1194

    accuracy                           0.97     12000
   macro avg       0.97      0.97      0.97     12000
weighted avg       0.97      0.97      0.97     12000
```

Clustering performance was evaluated using silhouette scores. The results indicate that both KMeans and KMedians produce comparable clustering quality on this dataset.