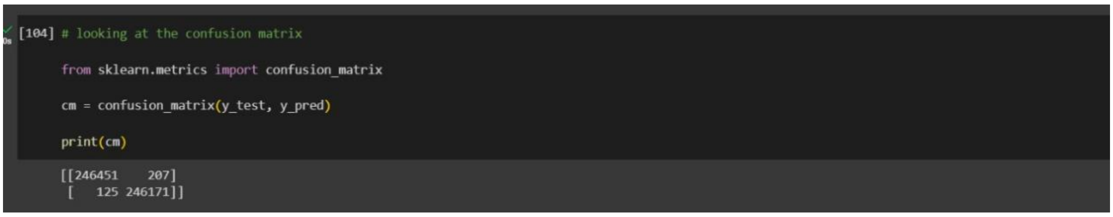
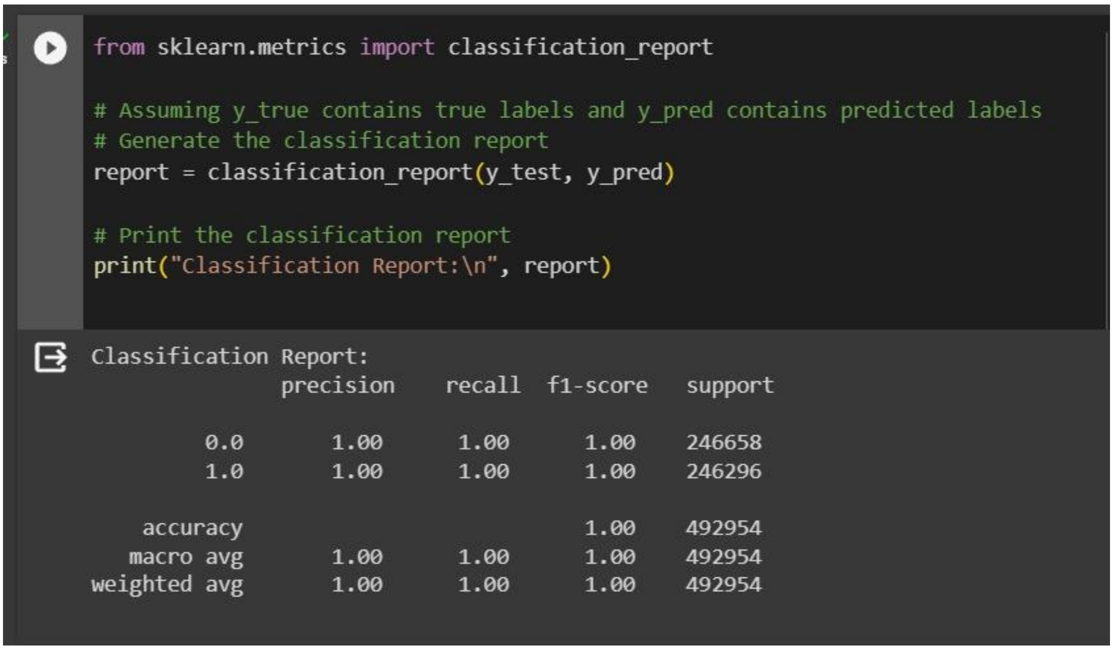
Confusion Matrix



```
[104] # looking at the confusion matrix

from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print(cm)
```

```
[[246451    207]
 [   125 246171]]
```

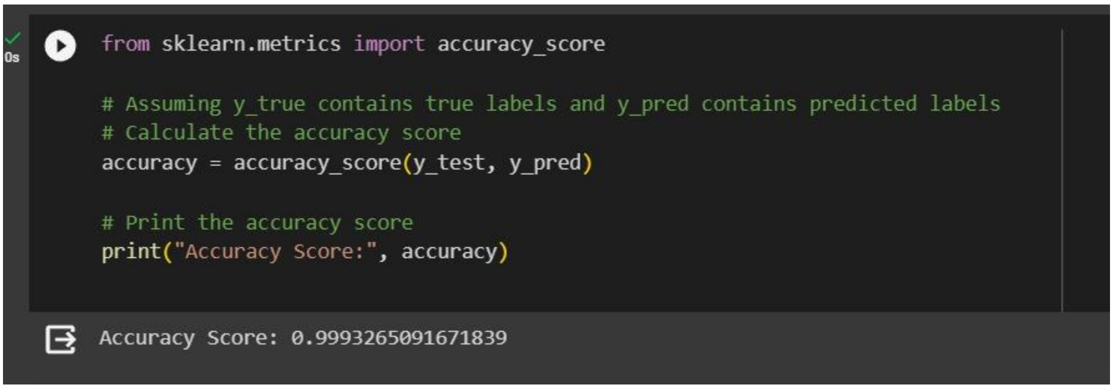
Classification Report



```
from sklearn.metrics import classification_report

# Assuming y_true contains true labels and y_pred contains predicted labels
# Generate the classification report
report = classification_report(y_test, y_pred)

# Print the classification report
print("Classification Report:\n", report)
```

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

         0.0       1.00      1.00      1.00    246658
         1.0       1.00      1.00      1.00    246296

    accuracy                           1.00    492954
   macro avg       1.00      1.00      1.00    492954
weighted avg       1.00      1.00      1.00    492954
```

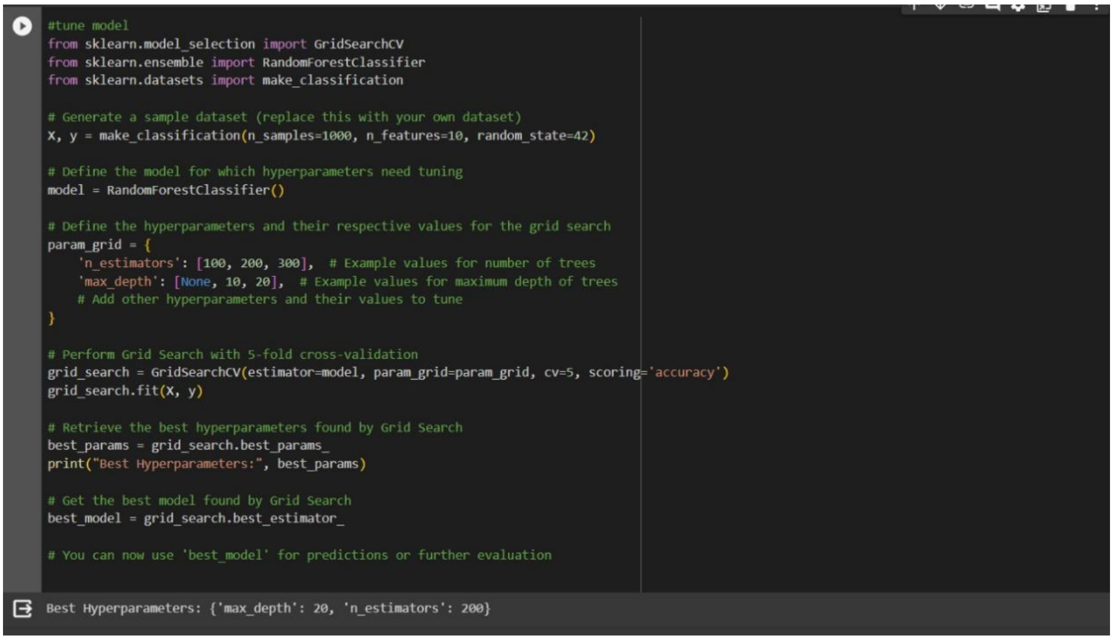
Accuracy Score



```
from sklearn.metrics import accuracy_score

# Assuming y_true contains true labels and y_pred contains predicted labels
# Calculate the accuracy score
accuracy = accuracy_score(y_test, y_pred)

# Print the accuracy score
print("Accuracy Score:", accuracy)
```

```
Accuracy Score: 0.9993265091671839
```

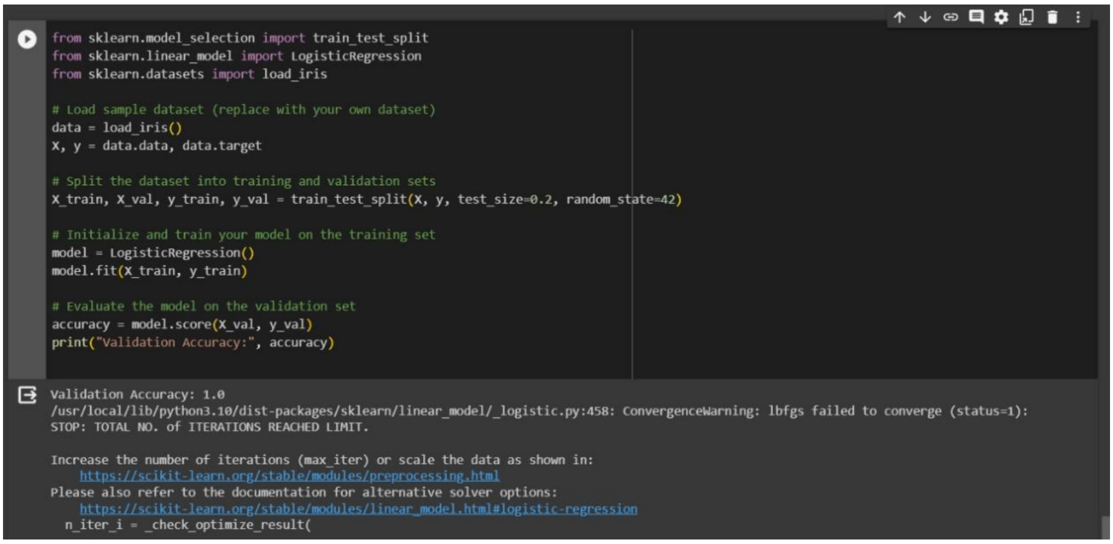
Hypertuning Tuning



```python
#tune model
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_classification

# Generate a sample dataset (replace this with your own dataset)
X, y = make_classification(n_samples=1000, n_features=10, random_state=42)

# Define the model for which hyperparameters need tuning
model = RandomForestClassifier()

# Define the hyperparameters and their respective values for the grid search
param_grid = {
    'n_estimators': [100, 200, 300],  # Example values for number of trees
    'max_depth': [None, 10, 20],  # Example values for maximum depth of trees
    # Add other hyperparameters and their values to tune
}

# Perform Grid Search with 5-fold cross-validation
grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X, y)

# Retrieve the best hyperparameters found by Grid Search
best_params = grid_search.best_params_
print("Best Hyperparameters:", best_params)

# Get the best model found by Grid Search
best_model = grid_search.best_estimator_

# You can now use 'best_model' for predictions or further evaluation
```

```
Best Hyperparameters: {'max_depth': 20, 'n_estimators': 200}
```

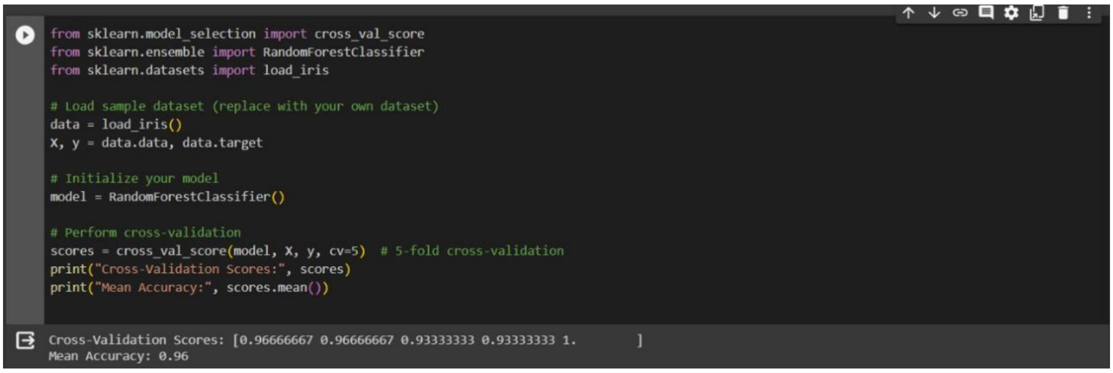
Holdout Validation



```python
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Load sample dataset (replace with your own dataset)
data = load_iris()
X, y = data.data, data.target

# Split the dataset into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train your model on the training set
model = LogisticRegression()
model.fit(X_train, y_train)

# Evaluate the model on the validation set
accuracy = model.score(X_val, y_val)
print("Validation Accuracy:", accuracy)
```

```
Validation Accuracy: 1.0
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

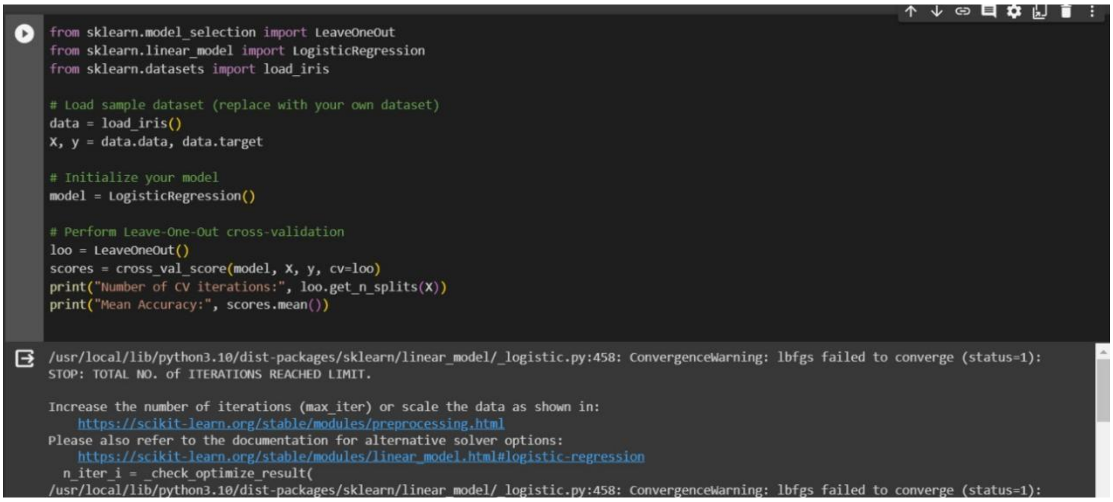
Cross Validation



```python
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris

# Load sample dataset (replace with your own dataset)
data = load_iris()
X, y = data.data, data.target

# Initialize your model
model = RandomForestClassifier()

# Perform cross-validation
scores = cross_val_score(model, X, y, cv=5)  # 5-fold cross-validation
print("Cross-Validation Scores:", scores)
print("Mean Accuracy:", scores.mean())
```

```
Cross-Validation Scores: [0.96666667 0.96666667 0.93333333 0.93333333 1.        ]
Mean Accuracy: 0.96
```

Leave-one-cross-out Validation



```python
from sklearn.model_selection import LeaveOneOut
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris

# Load sample dataset (replace with your own dataset)
data = load_iris()
X, y = data.data, data.target

# Initialize your model
model = LogisticRegression()

# Perform Leave-One-Out cross-validation
loo = LeaveOneOut()
scores = cross_val_score(model, X, y, cv=loo)
print("Number of CV iterations:", loo.get_n_splits(X))
print("Mean Accuracy:", scores.mean())
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed to converge (status=1):
```