15 points

great job!

Lopez_Monserrat_HW3

April 24, 2024

Welcome to the Machine Learning Assignment!

0.1 Objective

This assignment is designed to give you practical experience with a complete machine learning workflow, applying 10-fold cross validation, and running classification models on a prepared dataset.

0.2 Dataset Overview

You are provided with a dataset already split into training and test sets. Use the training set for developing and validating your models, and the test set for final evaluation.

0.3 Brief information about the dataset

This dataset contains data from a higher education institution on various variables related to undergraduate students, including demographics, social-economic factors, and academic performance, to investigate the impact of these factors on student dropout.

0.4 Task 1: Data Loading (2 mark)

Import necessary libraries and load the training and test datasets. Show the first five rows of each dataset to confirm loading, and use the info() method to review the details of the columns in each dataset.

[52]:
```
# Import the Libraries
import pandas as pd
from sklearn.model_selection import cross_val_score, train_test_split,␣
 ↪StratifiedKFold, cross_val_score, GridSearchCV, KFold
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report, accuracy_score
```

[15]:
```
#  Load the datasets

# Train data
```

```
train_data = pd.read_csv('/Users/monserratlopez/Library/CloudStorage/
 ↪GoogleDrive-lopezmonserrat.14@gmail.com/My Drive/4. Education/1_Master/
 ↪2_Hertie_MDS/2_SecondSemester/1_MachineLearning/1_Assignments/3_HW/
 ↪Lab2_Homework3/train_data.csv', sep=',')

# Test data
test_data = pd.read_csv('/Users/monserratlopez/Library/CloudStorage/
 ↪GoogleDrive-lopezmonserrat.14@gmail.com/My Drive/4. Education/1_Master/
 ↪2_Hertie_MDS/2_SecondSemester/1_MachineLearning/1_Assignments/3_HW/
 ↪Lab2_Homework3/test_data.csv', sep=',')

## Show the first five entries from both the training and testing datasets,
train_data.head()
```

[15]:

	Marital status	Application mode	Application order	Course	\
0	2	4	1	9	
1	1	8	1	10	
2	1	8	3	12	
3	1	8	1	6	
4	1	1	2	9	

	Daytime/evening attendance	Previous qualification	Nacionality	\
0	1	3	1	
1	1	1	1	
2	1	1	1	
3	1	1	1	
4	1	1	1	

	Mother's qualification	Father's qualification	Mother's occupation	…	\
0	1	4	4	…	
1	1	28	6	…	
2	1	14	8	…	
3	13	27	2	…	
4	1	14	5	…	

	Curricular units 2nd sem (credited)	Curricular units 2nd sem (enrolled)	\
0	0	5	
1	0	6	
2	0	8	
3	0	5	
4	0	5	

	Curricular units 2nd sem (evaluations)	\
0	5	
1	6	
2	11	
3	9	

```
4                                     5

   Curricular units 2nd sem (approved)  Curricular units 2nd sem (grade)  \
0                                    5                          13.80000
1                                    2                          11.50000
2                                    7                          12.94375
3                                    2                          13.50000
4                                    0                           0.00000

   Curricular units 2nd sem (without evaluations)  Unemployment rate  \
0                                               0               12.4
1                                               0               12.7
2                                               0               12.4
3                                               0                7.6
4                                               0               15.5

   Inflation rate   GDP  Target
0             0.5  1.79       1
1             3.7 -1.70       0
2             0.5  1.79       1
3             2.6  0.32       0
4             2.8 -4.06       0

[5 rows x 35 columns]
```

[16]:
```
test_data.head()
```

[16]:
```
   Marital status  Application mode  Application order  Course  \
0               1                 4                  1       9
1               1                 8                  1       9
2               1                 8                  3      12
3               1                 8                  1      12
4               5                12                  1       4

   Daytime/evening attendance  Previous qualification  Nacionality  \
0                           1                       2            1
1                           1                       1            1
2                           1                       1            1
3                           1                       1            1
4                           1                       1            1

   Mother's qualification  Father's qualification  Mother's occupation  …  \
0                       3                      27                    3  …
1                      13                      27                    6  …
2                       3                       3                   10  …
3                       1                      27                   10  …
4                       1                       1                    5  …
```

```
   Curricular units 2nd sem (credited)  Curricular units 2nd sem (enrolled)  \
0                                    0                                    5
1                                    0                                    5
2                                    0                                    8
3                                    0                                    8
4                                    0                                    6

   Curricular units 2nd sem (evaluations)  \
0                                       5
1                                      13
2                                       9
3                                       8
4                                       7

   Curricular units 2nd sem (approved)  Curricular units 2nd sem (grade)  \
0                                    0                          0.000000
1                                    1                         11.000000
2                                    8                         14.444444
3                                    8                         15.075000
4                                    0                          0.000000

   Curricular units 2nd sem (without evaluations)  Unemployment rate  \
0                                               0               15.5
1                                               0                9.4
2                                               0               13.9
3                                               0               13.9
4                                               0                8.9

   Inflation rate   GDP  Target
0             2.8 -4.06       0
1            -0.8 -3.12       0
2            -0.3  0.79       1
3            -0.3  0.79       1
4             1.4  3.51       0

[5 rows x 35 columns]
```

[17]:
```
## Use the info() method to review the details of the columns in each dataset.
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2904 entries, 0 to 2903
Data columns (total 35 columns):
 #   Column                                          Non-Null Count  Dtype
---  ------                                          --------------  -----
 0   Marital status                                  2904 non-null   int64
 1   Application mode                                2904 non-null   int64
```

```
 2   Application order                                2904 non-null   int64
 3   Course                                           2904 non-null   int64
 4   Daytime/evening attendance                       2904 non-null   int64
 5   Previous qualification                           2904 non-null   int64
 6   Nacionality                                      2904 non-null   int64
 7   Mother's qualification                           2904 non-null   int64
 8   Father's qualification                           2904 non-null   int64
 9   Mother's occupation                              2904 non-null   int64
 10  Father's occupation                              2904 non-null   int64
 11  Displaced                                        2904 non-null   int64
 12  Educational special needs                        2904 non-null   int64
 13  Debtor                                           2904 non-null   int64
 14  Tuition fees up to date                          2904 non-null   int64
 15  Gender                                           2904 non-null   int64
 16  Scholarship holder                               2904 non-null   int64
 17  Age at enrollment                                2904 non-null   int64
 18  International                                    2904 non-null   int64
 19  Curricular units 1st sem (credited)              2904 non-null   int64
 20  Curricular units 1st sem (enrolled)              2904 non-null   int64
 21  Curricular units 1st sem (evaluations)           2904 non-null   int64
 22  Curricular units 1st sem (approved)              2904 non-null   int64
 23  Curricular units 1st sem (grade)                 2904 non-null   float64
 24  Curricular units 1st sem (without evaluations)   2904 non-null   int64
 25  Curricular units 2nd sem (credited)              2904 non-null   int64
 26  Curricular units 2nd sem (enrolled)              2904 non-null   int64
 27  Curricular units 2nd sem (evaluations)           2904 non-null   int64
 28  Curricular units 2nd sem (approved)              2904 non-null   int64
 29  Curricular units 2nd sem (grade)                 2904 non-null   float64
 30  Curricular units 2nd sem (without evaluations)   2904 non-null   int64
 31  Unemployment rate                                2904 non-null   float64
 32  Inflation rate                                   2904 non-null   float64
 33  GDP                                              2904 non-null   float64
 34  Target                                           2904 non-null   int64
dtypes: float64(5), int64(30)
memory usage: 794.2 KB
```

[18]:
```
test_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 726 entries, 0 to 725
Data columns (total 35 columns):
 #   Column                                           Non-Null Count  Dtype
---  ------                                           --------------  -----
 0   Marital status                                   726 non-null    int64
 1   Application mode                                 726 non-null    int64
 2   Application order                                726 non-null    int64
 3   Course                                           726 non-null    int64
 4   Daytime/evening attendance                       726 non-null    int64
```

```
 5   Previous qualification                          726 non-null    int64
 6   Nacionality                                     726 non-null    int64
 7   Mother's qualification                          726 non-null    int64
 8   Father's qualification                          726 non-null    int64
 9   Mother's occupation                             726 non-null    int64
 10  Father's occupation                             726 non-null    int64
 11  Displaced                                       726 non-null    int64
 12  Educational special needs                       726 non-null    int64
 13  Debtor                                          726 non-null    int64
 14  Tuition fees up to date                         726 non-null    int64
 15  Gender                                          726 non-null    int64
 16  Scholarship holder                              726 non-null    int64
 17  Age at enrollment                               726 non-null    int64
 18  International                                   726 non-null    int64
 19  Curricular units 1st sem (credited)             726 non-null    int64
 20  Curricular units 1st sem (enrolled)             726 non-null    int64
 21  Curricular units 1st sem (evaluations)          726 non-null    int64
 22  Curricular units 1st sem (approved)             726 non-null    int64
 23  Curricular units 1st sem (grade)                726 non-null    float64
 24  Curricular units 1st sem (without evaluations)  726 non-null    int64
 25  Curricular units 2nd sem (credited)             726 non-null    int64
 26  Curricular units 2nd sem (enrolled)             726 non-null    int64
 27  Curricular units 2nd sem (evaluations)          726 non-null    int64
 28  Curricular units 2nd sem (approved)             726 non-null    int64
 29  Curricular units 2nd sem (grade)                726 non-null    float64
 30  Curricular units 2nd sem (without evaluations)  726 non-null    int64
 31  Unemployment rate                               726 non-null    float64
 32  Inflation rate                                  726 non-null    float64
 33  GDP                                             726 non-null    float64
 34  Target                                          726 non-null    int64
dtypes: float64(5), int64(30)
memory usage: 198.6 KB
```

0.5 Task 2: Separate Data and Labels (1 mark)

Separate features and target labels for both training and test datasets.

Last column "Target" has the labels for both training and test data.

Label 1= Graduate and Label 0 = Dropout

[20]:
```python
# Separate features and target labels for both training and test datasets.
# Define features and labels

# For the training data
X_train = train_data.drop(['Target'], axis=1)
y_train = train_data['Target'].astype(int)

# For test data
```

```
X_test = test_data.drop(['Target'], axis=1)
y_test = test_data['Target'].astype(int)
```

0.6 Task 3: Check for Missing Values (1 mark)

Check and handle any missing values in the training and test datasets.

[21]:
```
# For the training data
train_data.isna().sum() / len(train_data) * 100
```

[21]:
```
Marital status                                  0.0
Application mode                                0.0
Application order                               0.0
Course                                          0.0
Daytime/evening attendance                      0.0
Previous qualification                          0.0
Nacionality                                     0.0
Mother's qualification                          0.0
Father's qualification                          0.0
Mother's occupation                             0.0
Father's occupation                             0.0
Displaced                                       0.0
Educational special needs                       0.0
Debtor                                          0.0
Tuition fees up to date                         0.0
Gender                                          0.0
Scholarship holder                              0.0
Age at enrollment                               0.0
International                                   0.0
Curricular units 1st sem (credited)             0.0
Curricular units 1st sem (enrolled)             0.0
Curricular units 1st sem (evaluations)          0.0
Curricular units 1st sem (approved)             0.0
Curricular units 1st sem (grade)                0.0
Curricular units 1st sem (without evaluations)  0.0
Curricular units 2nd sem (credited)             0.0
Curricular units 2nd sem (enrolled)             0.0
Curricular units 2nd sem (evaluations)          0.0
Curricular units 2nd sem (approved)             0.0
Curricular units 2nd sem (grade)                0.0
Curricular units 2nd sem (without evaluations)  0.0
Unemployment rate                               0.0
Inflation rate                                  0.0
GDP                                             0.0
Target                                          0.0
dtype: float64
```

[23]:
```
# For test data
test_data.isna().sum() / len(test_data) * 100
```

[23]:
```
Marital status                                  0.0
Application mode                                0.0
Application order                               0.0
Course                                          0.0
Daytime/evening attendance                      0.0
Previous qualification                          0.0
Nacionality                                     0.0
Mother's qualification                          0.0
Father's qualification                          0.0
Mother's occupation                             0.0
Father's occupation                             0.0
Displaced                                       0.0
Educational special needs                       0.0
Debtor                                          0.0
Tuition fees up to date                         0.0
Gender                                          0.0
Scholarship holder                              0.0
Age at enrollment                               0.0
International                                   0.0
Curricular units 1st sem (credited)             0.0
Curricular units 1st sem (enrolled)             0.0
Curricular units 1st sem (evaluations)          0.0
Curricular units 1st sem (approved)             0.0
Curricular units 1st sem (grade)                0.0
Curricular units 1st sem (without evaluations)  0.0
Curricular units 2nd sem (credited)             0.0
Curricular units 2nd sem (enrolled)             0.0
Curricular units 2nd sem (evaluations)          0.0
Curricular units 2nd sem (approved)             0.0
Curricular units 2nd sem (grade)                0.0
Curricular units 2nd sem (without evaluations)  0.0
Unemployment rate                               0.0
Inflation rate                                  0.0
GDP                                             0.0
Target                                          0.0
dtype: float64
```

0.7 Task 4: Data Overview (1 marks)

Check the size of the dataset, including the number of training samples, test samples and features. Print these details.

[26]:
```
# Number of training samples and features
train_samples, train_features = X_train.shape
```

```
# Number of test samples and features
test_samples, test_features = X_test.shape

# Print the details
print("For the training dataset:")
print("Training samples:", train_samples)
print("Training features:", train_features)

print("\nFor the test dataset:")
print("Test samples:", test_samples)
print("Test features:", test_features)
```

```
For the training dataset:
Training samples: 2904
Training features: 34

For the test dataset:
Training samples: 726
Training features: 34
```

0.8 Task 5: 10-fold Cross Validation (1 marks)

Implement 10-fold cross validation on your standardized training data to validate the performance of your models.

[34]:
```
# Normalize the data
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

[37]:
```
model = LogisticRegression()

# Create a StratifiedKFold object with 10 folds
kf = KFold(n_splits=10, shuffle=True, random_state=42)

# Perform cross-validation and get the accuracy scores
scores = cross_val_score(model, X_train_scaled, y_train, cv=kf,␣
 ↪scoring='accuracy')

# Print the accuracy scores
print("Cross-validated Accuracy Scores:")
print(scores)
```

```
Cross-validated Accuracy Scores:
[0.90034364 0.9209622  0.91408935 0.91752577 0.90689655 0.91034483
 0.89655172 0.92758621 0.9137931  0.93103448]
```

0.9 Task 6: Model Implementation and Hyperparameter Tuning

Develop at least two different classification models.

0.9.1 Part A: First Classification Model (2 marks)

Model Implementation (1 mark): Implement your first classification model using any set of initial hyperparameters. Explain your choice of model and the initial parameters you selected.

Parameter Tuning (1 mark): Apply cross-validation to tune the parameters.

[45]:
```
# Apply Random Forest
rf = RandomForestClassifier(random_state=42)
rf.fit(X_train_scaled, y_train)
```

[45]:
```
RandomForestClassifier(random_state=42)
```

[50]:
```
# Grid search for hyperparameter tuning
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [None, 10],
    'min_samples_split': [2, 5]
}

cv_rf = GridSearchCV(estimator=rf, param_grid=param_grid, cv=10)
cv_rf.fit(X_train_scaled, y_train)
#cv_rf is the model after GridSearchCV
#has found the best parameters and refitted the model using these parameters on␣
 ↪the entire training set


# Best parameters
print("Best parameters:", cv_rf.best_params_)
```

```
Best parameters: {'max_depth': 10, 'min_samples_split': 2, 'n_estimators': 100}
```

[71]:
```
# Evaluate the model on the test set

# Predictions on the test set
y_pred_rf = cv_rf.predict(X_test_scaled)

# Performance metrics
print("Accuracy for Random Forest:", accuracy_score(y_test, y_pred_rf))
```

```
Accuracy for Random Forest: 0.90633608815427
```

0.9.2 Part B: Second Classification Model (2 marks)

Model Implementation (1 mark): Implement your second classification model with a different set of initial hyperparameters. Provide a rationale for your choice of model and the parameters.

Parameter Tuning (1 mark): Use cross-validation to refine the parameters.

[56]:
```
# Initialize and train the XGBoost classifier
xgb = XGBClassifier(use_label_encoder=False, eval_metric='logloss',␣
 ↪random_state=42)
xgb.fit(X_train_scaled, y_train)
```

[56]:
```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric='logloss',
              feature_types=None, gamma=None, grow_policy=None,
              importance_type=None, interaction_constraints=None,
              learning_rate=None, max_bin=None, max_cat_threshold=None,
              max_cat_to_onehot=None, max_delta_step=None, max_depth=None,
              max_leaves=None, min_child_weight=None, missing=nan,
              monotone_constraints=None, multi_strategy=None, n_estimators=None,
              n_jobs=None, num_parallel_tree=None, random_state=42, …)
```

[57]:
```
param_grid = {
    'n_estimators': [50, 100],
    'max_depth': [3, 5],
    'learning_rate': [0.01, 0.1]
}

# Initialize GridSearchCV and fit to find the best parameters
grid_search = GridSearchCV(xgb, param_grid, scoring='accuracy', cv=10,␣
 ↪verbose=1)
grid_search.fit(X_train_scaled, y_train)

# Print best parameters found by GridSearchCV
best_params = grid_search.best_params_
print("Best parameters:", best_params)
```

```
Fitting 10 folds for each of 8 candidates, totalling 80 fits
Best parameters: {'learning_rate': 0.1, 'max_depth': 3, 'n_estimators': 100}
```

[72]:
```
# Predictions on the test set
y_pred_xgb = grid_search.predict(X_test_scaled)

# Performance metrics
print("Accuracy for XGBoost classifier:", accuracy_score(y_test, y_pred_xgb))
```

```
Accuracy for XGBoost classifier: 0.9173553719008265
```

0.10 Task 7: Summarize the comparison of the two models in terms of accuracy and confusion matrices(1 mark). Plot the ROC-AUC Curve (1 mark).

[73]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc
import matplotlib.pyplot as plt

# Calculate accuracy for Random Forest
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Accuracy for Random Forest:", accuracy_rf)

# Calculate accuracy for XGBoost
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print("Accuracy for XGBoost:", accuracy_xgb)

# Confusion matrix for Random Forest
cm_rf = confusion_matrix(y_test, y_pred_rf)
print("Confusion matrix for Random Forest:")
print(cm_rf)

# Confusion matrix for XGBoost
cm_xgb = confusion_matrix(y_test, y_pred_xgb)
print("Confusion matrix for XGBoost:")
print(cm_xgb)
```

```
Accuracy for Random Forest: 0.90633608815427
Accuracy for XGBoost: 0.9173553719008265
Confusion matrix for Random Forest:
[[220  43]
 [ 25 438]]
Confusion matrix for XGBoost:
[[221  42]
 [ 18 445]]
```

nice work!

[76]:
```
# Plot the ROC-AUC Curve
from sklearn.metrics import roc_curve, auc
import matplotlib.pyplot as plt

# Compute predicted probabilities for Random Forest and XGBoost
y_pred_rf = cv_rf.predict_proba(X_test_scaled)[:, 1]
y_pred_xgb = grid_search.predict_proba(X_test_scaled)[:, 1]

# Compute ROC curve and ROC area for Random Forest
fpr_rf, tpr_rf, _ = roc_curve(y_test, y_pred_rf)
roc_auc_rf = auc(fpr_rf, tpr_rf)

# Compute ROC curve and ROC area for XGBoost
fpr_xgb, tpr_xgb, _ = roc_curve(y_test, y_pred_xgb)
roc_auc_xgb = auc(fpr_xgb, tpr_xgb)
```

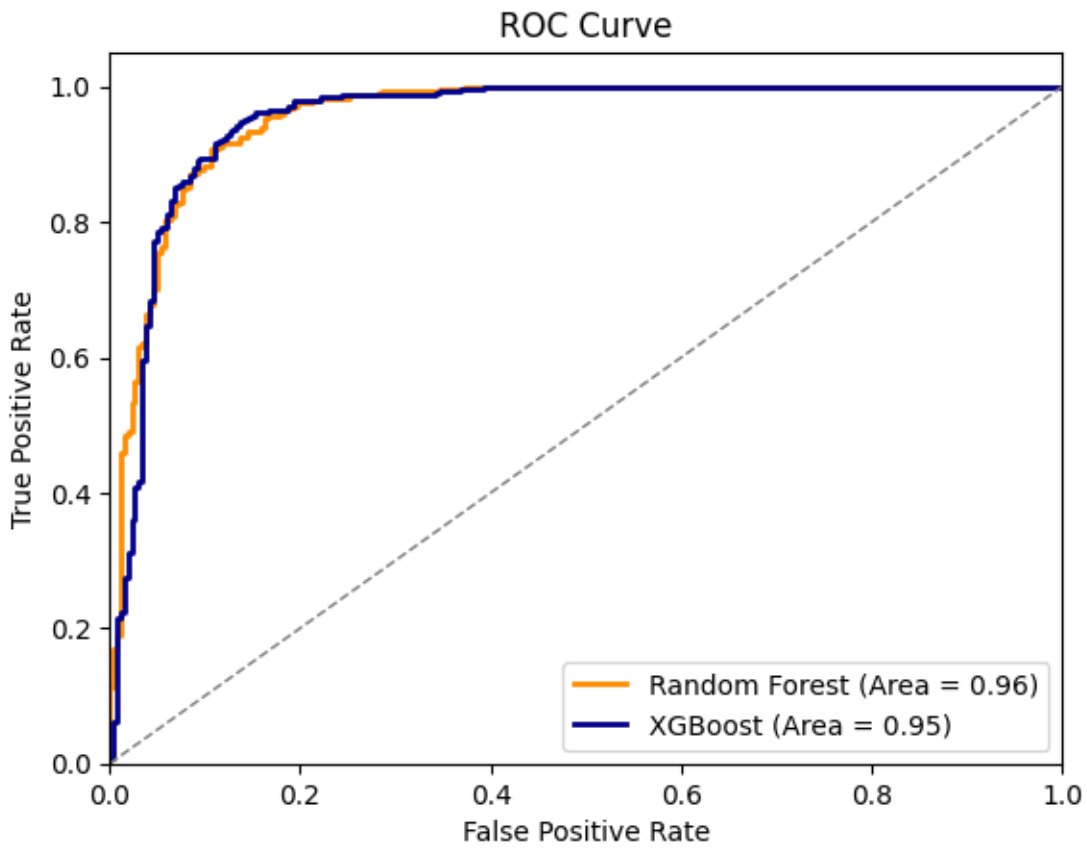
```
# Plot ROC curve for Random Forest
plt.figure()
plt.plot(fpr_rf, tpr_rf, color='darkorange', lw=2, label='Random Forest (Area =␣
 ↪%0.2f)' % roc_auc_rf)

# Plot ROC curve for XGBoost
plt.plot(fpr_xgb, tpr_xgb, color='darkblue', lw=2, label='XGBoost (Area = %0.
 ↪2f)' % roc_auc_xgb)

# Plot ROC curve for random guessing (diagonal line)
plt.plot([0, 1], [0, 1], color='gray', lw=1, linestyle='--')

# Set plot labels and title
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc="lower right")

# Show plot
plt.show()
```



0.11 Task 8: Friendly Competition for Highest Accuracy (3 marks)

Let's make this interesting! Participate in a friendly competition with your classmates to achieve the highest accuracy on the test set. To add a little extra fun, we will maintain a leaderboard on the classroom whiteboard, updating it regularly to showcase the top performances.

This is a great opportunity to learn from each other, push your skills to the limit, and maybe even earn some bragging rights!

Best of luck, and let's see what you can achieve!

[77]:
```
### print("Test set accuracy for the chosen model:", accuracy_score(y_test,␣
↪y_pred_xgb)) # y_pred has the test labels predicted by your model
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[77], line 1
----> 1 print("Test set accuracy for the chosen model:",␣
↪accuracy_score(y_test, y_pred_xgb)) # y_pred has the test labels predicted by␣
↪your model
```

```
File ~/miniconda3/envs/ML/lib/python3.10/site-packages/sklearn/utils/
 ↪_param_validation.py:213, in validate_params.<locals>.decorator.<locals>.
 ↪wrapper(*args, **kwargs)
    207 try:
    208     with config_context(
    209         skip_parameter_validation=(
    210             prefer_skip_nested_validation or global_skip_validation
    211         )
    212     ):
--> 213         return func(*args, **kwargs)
    214 except InvalidParameterError as e:
    215     # When the function is just a wrapper around an estimator, we allow
    216     # the function to delegate validation to the estimator, but we␣
 ↪replace
    217     # the name of the estimator by the name of the function in the erro
    218     # message to avoid confusion.
    219     msg = re.sub(
    220         r"parameter of \w+ must be",
    221         f"parameter of {func.__qualname__} must be",
    222         str(e),
    223     )

File ~/miniconda3/envs/ML/lib/python3.10/site-packages/sklearn/metrics/
 ↪_classification.py:213, in accuracy_score(y_true, y_pred, normalize,␣
 ↪sample_weight)
    147 """Accuracy classification score.
    148
    149 In multilabel classification, this function computes subset accuracy:
   (…)
    209 0.5
    210 """
    212 # Compute accuracy for each possible representation
--> 213 y_type, y_true, y_pred = _check_targets(y_true, y_pred)
    214 check_consistent_length(y_true, y_pred, sample_weight)
    215 if y_type.startswith("multilabel"):

File ~/miniconda3/envs/ML/lib/python3.10/site-packages/sklearn/metrics/
 ↪_classification.py:94, in _check_targets(y_true, y_pred)
     91     y_type = {"multiclass"}
     93 if len(y_type) > 1:
---> 94     raise ValueError(
     95         "Classification metrics can't handle a mix of {0} and {1}␣
 ↪targets".format(
     96             type_true, type_pred
     97         )
     98     )
    100 # We can't have more than one value on y_type => The set is no more␣
 ↪needed
```

```
    101 y_type = y_type.pop()

ValueError: Classification metrics can't handle a mix of binary and continuous␣
 ↪targets
```

[ ]: