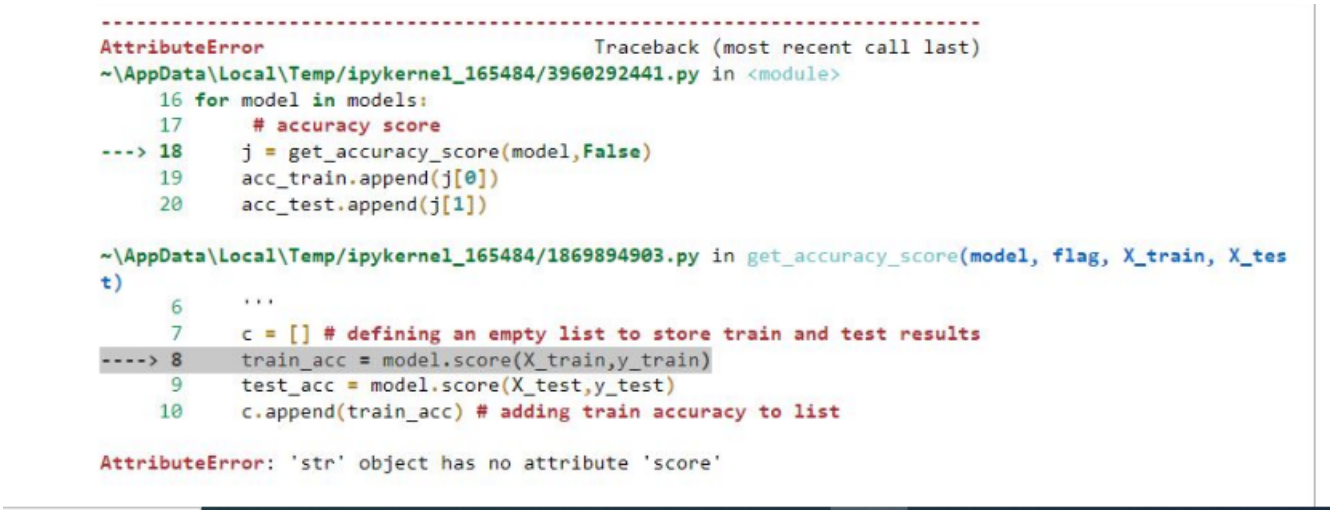
← Go Back to Capstone Project

Course Content

FAQs - Loan Default Prediction

1. Getting the following error while running the code mentioned in the below screenshot. How to resolve this error?

```
AttributeError: 'str' object has no attribute 'score'
```



```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_165484/3960292441.py in <module>
     16 for model in models:
     17     # accuracy score
---> 18     j = get_accuracy_score(model,False)
     19     acc_train.append(j[0])
     20     acc_test.append(j[1])

~\AppData\Local\Temp/ipykernel_165484/1869894903.py in get_accuracy_score(model, flag, X_train, X_tes
t)
      6     '''
      7     c = [] # defining an empty list to store train and test results
----> 8     train_acc = model.score(X_train,y_train)
      9     test_acc = model.score(X_test,y_test)
     10     c.append(train_acc) # adding train accuracy to list

AttributeError: 'str' object has no attribute 'score'
```

This error occurs when we try to access an attribute that doesn't exist for string objects. To resolve the error, make sure the value is of the expected type before accessing the attribute.

You should pass model objects, i.e., variable names instead of strings in the model's list

Please refer to the below code for a better understanding:

```
models = [dtree, dtree_estimator, rf, rf_wt, rf_estimator]
```

2. Can you please explain the below line of code?

```
y_scores_logreg[:, 1]
```

Classifier models like logistic regression do not actually output class labels (like "0" or "1"), they give output as the probability. These probabilities tell you the likelihood that the input sample belongs to class 1. However, you still need to choose a probability threshold so that the algorithm can convert the probability into a class.

If you choose a threshold of 0.5 (the default threshold), then all input samples with calculated probabilities greater than 0.5 will be assigned to class 1, and samples with probabilities less than 0.5 will be assigned to class 0. If you choose a different threshold, then you get a different number of samples assigned to the positive and negative class, and therefore different precision and recall scores.

So, by varying the threshold we have different prediction results. In many problems, a much better result may be obtained by adjusting the threshold. The optimal threshold can be determined by the precision_recall_curve.

Please refer to the below source for a better understanding.

The predictions were stored in the y_scores_logreg object, which will return the probabilities for each class of all samples in a test set, e.g., two numbers, one for each of the two classes, in a binary classification problem. The probabilities for the positive class can be retrieved as the second column in this array of probabilities by this code, i.e., y_scores_logreg[:,1].

3. The training and the test outputs for the confusion matrix were all 100% accurate. Why is this the case?

This is because you are dropping other variables as well other than 'BAD'. Please kindly refer to the below code while separating the target variable and other variables:

```
X = df.drop(['BAD'],axis=1)
X = pd.get_dummies(X,drop_first=True)
y = df['BAD']
```

4. Getting confused about the class labeling in the confusion matrix. Can you please explain?

The model can make wrong predictions in the following ways:

- Predicting a customer will default but the customer doesn't default in reality (False Positive)
- Predicting a customer will not default but the customer defaults in reality (False Negative)

Which loss is more expensive/damaging for the organization?

Predicting that customers will not default but defaults in reality (False Negative) is more expensive as this will lead to a huge loss for the bank if customers do not pay the loan back to the bank. Therefore, banks would want to capture as many customers as possible who would not be able to repay the loan (defaulters).

How to reduce this loss, i.e., need to reduce False Negatives?

Bank wants Recall to be maximized, the greater the Recall higher the chances of minimizing false negatives. Hence, the focus should be on increasing Recall or minimizing the false negatives or in other words identifying the true positives (i.e., Class 1) so that the bank can minimize the BAD loans.

The description of the target variables is as follows:

BAD: 1 = Client defaulted on loan, 0 = loan repaid, i.e., 1 is defaulting and 0 is not defaulting.

We want to predict who is going to default. So, considering from the top left cell and proceeding in the clockwise direction, the confusion matrix indicates:

1) True Negative (Top Left Cell): Model correctly predicts that the customer will not default

2) False Positive (Top Right Cell): Model wrongly predicts that the customer will default

3) True Positive (Bottom Right Cell): Model correctly predicts that the customer will default

4) False Negative (Bottom Left Cell): Model wrongly predicts that the customer will not default

5. Do you have any resources with some guidance for writing better observations, summaries, and recommendations?

The conclusions and recommendations should be totally based on the key findings of your entire project. It should be based on the insights, the model outputs, and the observations you have made during the analysis.

You can refer to MLS and/or practice case studies to see how to write observations and summaries.

Basically, observations and summaries are the important findings from EDA, your analysis drawn from it, and the steps taken to solve the problem.

6. Facing difficulty with the code while treatment of outliers getting NA values everywhere after trying to apply the defined function?

```
df
```

590]:

	BAD	LOAN	MORTDUE	VALUE	REASON	JOB	YOJ	DEROG	DELINQ	CLAGE	NINQ	CLNO	DEBTINC
0	1	NaN	NaN	NaN	HomeImp	Other	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	1	NaN	NaN	NaN	HomeImp	Other	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	1	NaN	NaN	NaN	HomeImp	Other	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

This issue might be generating because in the treat_outliers() function, you are calculating the 25th and the 75th quantile using the whole dataframe 'df', however, you must calculate them for each column individually. For example, to calculate the 25th quantile, you can use the below code:

```
df[col].quantile(q = 0.25)
```

instead of:

```
df.quantile(q = 0.25)
```

Also, while imputing missing values, you might be using the dataframe 'data' while you must use the dataframe 'df' as we need to impute missing values on the most recent data where we treated outliers.

7. Which attributes to consider while splitting the data? Should we use regular attributes or scaled attributes?

Please pass the scaled data for train_test_split. Let's say X is a combination of independent variables and after scaling X, if you are storing the scaled data in X_scale, then you have to pass X_scale for train_test_split.

8. When do we use random search instead of grid search? What is the difference between the two?

A method for finding the optimal parameter values in a grid of parameters is called GridSearchCV. Essentially, it is a cross-validation method. **GridSearchCV tries all the combinations** of the values passed in the dictionary and evaluates the model for each combination. Predictions are made after extracting the optimal parameter values.

RandomizedSearchCV is similar to GridSearchCV in that we use **random hyperparameter combinations** to find the optimal hyperparameter values to build the model. The number of combinations to use is passed to the algorithm as a parameter (n_iter). Unlike grid search, RandomizedSearchCV does not consider all values, and the values that are considered are chosen at random.

For example, if the distribution has 500 values and we enter n_iter=50, the random search will randomly sample 50 combinations to validate.

The primary advantage of RandomizedSearchCV is faster computation. It can be two, three, or four times faster than GridSearchCV depending on the n_iter used. However, the higher the n iter, the slower RandomizedSearchCV will be and the algorithm will be more similar to GridSearchCV.

9. In the below line of code:

```
scorer=metrics.make_scorer(recall_score, pos_label = 1)
```

What does "pos_lable=1" mean?

pos_label is an argument of scikit-learn's make_scorer; its purpose is to indicate which label needs to be considered as the positive one (i.e., pos_label is used to specify the label of the positive class) and, if not given explicitly, it assumes the class with label 1 is a positive class.

Kindly refer to this source for a better understanding.

10. What is the F1-Score?

F1-Score is used when the False Negatives and False Positives both are crucial to our problem. It elegantly sums up the predictive performance of a model by combining two otherwise competing metrics, i.e., precision and recall using the below formula:

```
F1 = 2 * (precision * recall) / (precision + recall)
```

Higher the F1 score, the better the performance of the model. F1 scores can range from 0 to 1, with 1 representing a model that perfectly classifies each observation into the correct class and 0 representing a model that is unable to classify any observation into the correct class.

11. Does the fact that the dataset has 4771 out of 5960 as non-defaulters make the data biased? If yes, how can this be fixed to provide an unbiased model?

Data imbalance will be common in a real-world scenario. We can create good models on the imbalanced data as well. This can be achieved by giving different weights to both the majority and minority classes. The difference in weights will influence the classification of the classes during the training phase. The whole purpose is to penalize the misclassification made by the minority class by setting a higher class weight and at the same time reducing weight for the majority class.

Note: There is a threshold to which you should increase and decrease the class weights for the minority and majority classes. If you give very high-class weights to the minority class, chances are the algorithm will get biased towards the minority class, and it will increase the errors in the majority class.

12. Getting the below error while scaling the attributes. How to resolve this error?

```
ValueError: could not convert string to float
```

The error occurs if you try to convert a string to a float that contains invalid characters. Please convert the 'LOAN' variable, which is in string datatype, into numerical datatype using encoding techniques. After transforming the variable, try to standardize the data using the standard scaler.

Happy Learning!

‹ Previous

Next ›