WEEK 11 DELIVERABLES

Team Member's Details

Group Name: Data Bank

Tevfik SASTIM, talfik22@gmail.com, Turkey, Self- Employed, Data Science

Marelly Escelade, idemany1@gmail.com, Peru, Self- Employed, Data Science

Problem Statement:

ABC Bank wants to sell it's term deposit product to customers and before launching the product they want to develop a model which help them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution).

We are now working on this problem through the following steps to find best solution for our customer to give the bank ability to better target its customers and personalize the marker plan.

Github Repo Link

Github Repo link: https://github.com/talfik2/Data-Glacier-Data-Bank-Bank-Marketing-Group-Project

Link for Step 5: https://github.com/talfik2/Data-Glacier-Data-Bank-Bank-Marketing-Group-Project/tree/Step-5-Model-evaluation

EDA Presentation and proposed modeling technique

This step is completed by Tevfik SASTIM. In this step,

Model's measure metric is set as F1 Score as our data is highly imbalanced.

Moreover, model is evaluated by Bias – Variance Tradeoff,

BIAS - VARIANCE TRADE OFF

```
In [61]: # F1 Score for Training Set
         y_pred = pipeline.predict(X_train)
         print("F1 Score for Training Set is: ", (f1_score(y_pred, y_train, average = "weighted")))
         # Pipeline'ımız için Mean Absolute Error(Test Set için)
         y_pred = pipeline.predict(X_test)
         print("F1 Score for Test Set is: ", (f1_score(y_pred, y_test, average = "weighted")))
```

```
F1 Score for Training Set is:  0.9157562507824775
F1 Score for Test Set is:  0.8890774818079843
```

- F1 Scores for Training & Test Set's are pretty good as we consider the best F1 Score = 1.
- We can not say that our model is overtrained because training set F1 Score is very close to 1.0 & test set F1 Score is slighly lower than training set F1 Score.
- We also can not say that our model is undertrained since training & test F1 Scores are pretty good.
- Thus, I can say that our model performs optimally.

And it is validated by Traditional Validation, K – fold Cross Validation & Randomized Search CV.

- Thus, I can say that our model performs optimally.

MODEL VALIDATION

Traditional Validation

```
In [62]: X_temp, X_test, y_temp ,y_test = train_test_split(
             banking.drop(columns=['y','duration']),
             banking["y"],
             test_size=0.20,
             random_state=42,
             stratify=banking["y"])
```

```
In [64]: X_train, X_val, y_train ,y_val = train_test_split(
             #X
             X_temp,
             #y
             y_temp,
             # % 25 of the validation set = % 20 of general data
             test_size=0.25,
             random_state=42)
         pipeline.fit(X_train, y_train)
         y_pred = pipeline.predict(X_val)
         print("F1 Score Before Traditional Validation: 0.8890774818079843")
         print("F1 Score After Traditional Validation: ", (f1_score(y_pred, y_val, average = "weighted")))
```

```
F1 Score Before Traditional Validation: 0.8890774818079843
F1 Score After Traditional Validation:  0.8865318712776289
```

- It is very slightly lower before the validation. So this is what traditional validation does, it makes either none or sightly lower difference.

Cross Validation

```
In [72]: from sklearn.model_selection import cross_val_score
         from sklearn.metrics import make_scorer
         f1 = make_scorer(f1_score)
         cv_results = cross_val_score(pipeline,X_train, y_train, cv = 5, scoring = f1)
         print("F1 Score Before Traditional Validation: 0.8890774818079843")
         print("F1 Score After Traditional Validation: ", cv_results.mean())
```

```
F1 Score Before Traditional Validation: 0.8890774818079843
F1 Score After Traditional Validation:  0.29197068209826843
```

- Here, you can see the importance of the hyperparameters. As our data is inbalanced, it performs bad except the average = "weighted"

Randomized SeachCV

```
In [77]: from sklearn.model_selection import RandomizedSearchCV
         import numpy as np
         dtc = DecisionTreeClassifier()
         # Optimize etmek istediğimiz dtr hyperparameterelerini belirtelim
         trans_param_grid = {"max_depth":np.arange(1,100,1),
                             "min_samples_leaf":np.arange(1,100,1),
                             "min_samples_split":np.arange(1,100,1),
                             "random_state": np.arange(1,200,1)}

         randomized_accuracy = RandomizedSearchCV(estimator=dtc, param_distributions = trans_param_grid,
                                                  # n_iter = 40 performs as well as GridSeachCV with lesser time
                                                  n_iter = 40, scoring = "f1", cv = 5,
                                                  verbose = 1)
         randomized_accuracy.fit(X_train, y_train)
         print("Best Params are: ", randomized_accuracy.best_params_)
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
C:\Users\talfi\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py:610: FitFailedWarning: Estimator fit failed. The score
on this train-test partition for these parameters will be set to nan. Details:
Traceback (most recent call last):
  File "C:\Users\talfi\anaconda3\lib\site-packages\sklearn\model_selection\_validation.py", line 593, in _fit_and_score
```

Final Recommendation

Everything seems to be in order. If you have any feedback, feel free to contact us.