AIML Online

Frequently Asked Questions in Problem Statement

Course: Ensemble Techniques

PART - A [30 Marks]

** Direct or Self-explanatory questions are not covered in this FAQ.*

1. Data Understanding & Exploration:

1 D. Verify if all the columns are incorporated in the merged DataFrame by using a simple comparison Operator in Python. [1 Marks]

→ *Compare the columns of the original dataframes and the merged ones along with the dimensions of the dataframes (no. of columns & no. of rows).*

2. Data Cleaning & Analysis: [5 Marks]

2 A. Impute missing/unexpected values in the DataFrame. [2 Marks]

→ *Unexpected values are nothing but irrelevant values of those columns or any other special characters or empty spaces.*

2 B. Make sure all the variables with continuous values are of 'Float' type. [2 Marks]

→ *Convert all numeric variables to float and verify.*

2 C. Create a function that will accept a DataFrame as input and return pie-charts for all the appropriate Categorical features. Clearly show percentage distribution in the pie-chart. [4 Marks]

→ *You have to define a function and within that create a loop to select columns that are object type. Exclude unnecessary columns. Calculate the % within the loop and plot a pie chart within the loop.*

Hint: use value_counts to calculate the % of distribution for pie chart

3. Model building and Improvement: [10 Marks]

3. A. Train a model using XGBoost. Also print best performing parameters along with train and test performance. [5 Marks]

B. Improve performance of the XGBoost as much as possible. Also print best performing parameters along with train and test performance. [5 Marks]

→ For A. You have to build a basic XGBoost Classifier Models with a range of random parameter values. Here, you can use the random search CV and print the best parameters.

For B. You can build a new model and tune the hyper parameters further based on the best parameters from the previous step and display the best ones. Here, you can use GridSearch search.

Hint: Please explore the RandomizedSearchCV function and add the XGB Classifier model in the function along with the set of parameter values. You can define parameter values as a set and use them in the RandomizedSearchCV. Eg. 'learning rate' : uniform(0.07, 0.12), 'n_estimators': randint(20, 70).

Similarly, you can perform GridSearchCV for the second question (B) and tune the parameters based on the best parameters from 1st step (A)

Reference link - https://grabngoinfo.com/hyperparameter-tuning-for-xgboost-grid-search-vs-random-search-vs-bayesian-optimization/

PART - B [30 Marks]

→ *Part-B involves a little bit of research. The expectation is not to use pipelines instead to build a manual workflow as per given instructions.*

You can refer to this link to understand how the functions and loops works to build a workflow - https://www.kdnuggets.com/2017/05/machine-learning-workflows-python-scratch-part-1.html

On top of the above reference, please build your own workflow as per hints given below:

Create separate functions for every step individually.

Eg: for pre-processing, you can define a function which can take Independent variables and dependent variables as input. Now you can segregate the numeric and categorical values by its data type and impute missing values (you can use functions that you used in part 1 of the project). Similarly, you can do outlier treatment or normalization, or dropping unnecessary features etc and those steps should be within the pre-processing function that you created. Then the function can return the pre-processed dataset as the final output.

Now, you can use this pre-processed dataset for building models in another user-defined function, where you can encode the variables, split the dataset, build multiple models, and print model performance. Then, the base models and the performance metrics of those models can be captured and transformed as a simple data frame (with train accuracy, test accuracy, precision, recall etc.)

You can use the performance data frame in a separate user defined function to pick the best performing model based on the metrics and run the best model with the training dataset. Then the models can be saved for future use using pickle (You can explore this pickle, it's a simple function)

Modularization is building separate user-defined functions as mentioned above and it is termed an Industry approach. Traditional approach is the same as how part 1 of the project is done. If the traditional approach is carried out in part 2, you will get 20 out of 30 marks.

Example: The below snippet explains what can be inside the pre-processing function

```
# Function to preprocess the data

def preprocess_data(path_to_csv,target_feature):

df = pd.read_csv(path_to_csv)

# Impute missing values (you can use loops and conditions imputing missing values

based on dtypes)

<code for imputing missing values>

# Impute duplicate values (you can drop duplicate values)

<code for identifying and dropping duplicate values>

#Dropping unnecessary features

<Code for dropping unnecessary features>

return df,target_feature
```

Similarly, you can build other functions for splitting the data, initiating and training models, measuring model performance, pickling the model etc. as given in the problem statement. Finally, compile all functions inside a main function and call that main function with necessary parameters which results in printing the model performance of each model.