**Milestone 3**

**Feature Selection, parameter tuning and model training**

**Data description:**

1. **Find the generated data file from the milestone 1 -** 'new\_finance.csv'**. We will use this preprocessed data file.**
2. **Run the Following Code in Jupyter Notebook and produce the outputs wherever necessary. Answer all questions in your ipynb python jupyter file itself (you can use markdown cells as well wherever necessary) and submit the pdf file of your code run with results. (Failing to not submit the pdf file will result in 100% deduction in marks). For this mile stone you need to submit 2 separate pdf of your run code- one for feature selection and one for cross validation parameter tuning.**

**Step 1# Perform the Feature Selection-Information gain - mutual information (MI). Here write the code of your choice to perform this step. Here is the little help to get started.**

**import pandas as pd**

dataset = pd.read\_csv('new\_finance.csv', index\_col=0)

print(dataset)

*# drop categorical duplicate variables -* **Here run the following code.**

dataset=dataset.drop(['Loan\_SubType','Loan\_Type','Loan\_Status' ,'Borrower\_Type','Applicant\_State','Type

\_ofVehicle','Loan\_Class','Loan\_Tier'], axis=1)

*# Train test split -* **Here run the following code.**

**from sklearn.model\_selection import** train\_test\_split

X\_train,X\_test,y\_train,y\_test=train\_test\_split(dataset.drop(labels=['Loan\_SubType\_Cat'], axis=1), dataset['Loan\_SubType\_Cat'], test\_size=0.3,

random\_state=0)

**# Let’s import libraries for Information gain - mutual information (MI) feature selection – Here you have to fill the blanks by library and method names prior to run the code.**

**from sklearn.-------------- import ------------------**

*# determine the mutual information*

mutual\_info = ----------------(X\_train, y\_train)

***Step 2****# show all the MI score for each variable in the data – You can run your own suitable code but* **Here is the Skelton for this step to run the code.**

mutual\_info = pd.Series(mutual\_info)

mutual\_info.index = X\_train.columns

mutual\_info.sort\_values(ascending=**False**)

**After this you have received a list of all variables with their scores. Please answer the following questions in your jupyter file itself.**

**a. what are the top 15 variables according to you that you want to select for further prediction task and why.**

**Step 3# Now lets perform the cross validation for parameter tuning. Please create new ipynb file for this task.**

**Step 3.1*# Lets tune the parameters for*** *KNN Hyperparameter Tuning-* **write your own code to do this but include following information of parameters of KNN:-**

**Use grid search CV for cross validation.**

***consider K values between 3 and 7 and p values of 1 (Manhattan), 2 (Euclidean), and 5 (Minkowski).***

***Use these parameter values in GridSearchCV:- cv=5,verbose=1,scoring='accuracy',return\_train\_score=True***

***If you go to sklearn KNN classifier page you will find many parameters but we will use two important ones – K (n\_* *neighbors') and P***

***Step 3.2*** *write a code getting KNN* ***best parameter values*** *for n\_neighbor and p using the train dataset.*

***Step 3.3*** *Write a code to fit again KNN on final tuned best parameter’s values that you have just received above.*

**Step 3.4** write a code to show the performance measures as classification report for the KNN.

**This task is very important to perform otherwise there will be 60% marks deduction.**

**🡪 Same as above, now perform the cross validation for the parameter tuning of decision tree (DT) and SVC (Support vector machine) ML classification techniques and fit these (DT and SVC) on the best tuned parameter values and generate the classification reports for both. (the steps are the same as we did for the KNN, you just need to pay attention on the parameters of SVC and DT). If you have watched the lecture videos then you know what are the important parameters for DT and SVC, you just need to tune them not all. Run your code and generate the results. In a separate cell of your code as a markdown text Tell me- according to you which technique performed better and have you observe issues like overfitting/underfitting/sample unbalancing. provide the pdf of your code.**

-----------\*\*\*\*\*\*\*\*\*\*\*\*\*------------------