TITLE

**Predict Blood Donation**

**Dataset**

**Blood Transfusion Service Center , transfusion.csv**

**Project Description –**

**Forecasting blood supply is a serious and recurrent problem for blood collection managers.**

**In this Project, you will work with data collected from the donor database of Blood Transfusion**

**Service Center. The dataset, obtained from the Machine Learning Repository, consists of a**

**random sample of 748 donors. Your task will be to predict if a blood donor will donate within a given**

**time window. You will look at the full model-building process: from inspecting the dataset to using the library to automate your Machine Learning pipeline. To complete this Project, you need to know some Python, pandas, and logistic regression.**

**Task 1: Instructions**

**Inspect the file that contains the dataset.**

**Print out the first 5 lines from datasets/transfusion.data using the head shell**

**command.**

**Make sure to first read the narrative for each task in the notebook on the right before reading the**

**more detailed instructions here. To complete this Project, you need to know some Python, pandas,**

**and logistic regression. We recommend one is familiar with the content.**

**To run a shell command in a notebook, you prefix it with: - ! , e.g. !ls**

**will list directory contents.**

**Task 2: Instructions**

**Load the dataset.**

**Import the pandas library Load the transfusion.data file from datasets/transfusion.data and assign it to the transfusion variable. Display the first rows of the DataFrame with the head() method to verify the file was loaded correctly If you print the first few rows of data, you should see a table with only 5 Column.**

**Task 3: Instructions**

**Inspect the DataFrame's structure. Print a concise summary of the transfusion Data Frame with the info() method. Data Frame’s info()method prints some useful information about a DataFrame:-**

* **Index type**
* **Column type**

**Task 4: Instructions**

**Rename a column.**

**Rename whether he/she donated blood in March 2007 to target for brevity Print the first 2 rows of the DataFrame with the head() method to verify the change was done correctly By setting the in place parameter of the rename() method to True, The transfusion DataFrame is changed in-place, i.e., the transfusion variable will now point to the updated DataFrame as you'll verify by printing the first 2 rows.**

**Task 5: Instructions**

**Print target incidence. Use value\_counts() method on transfusion.target column to print target incidence proportions, setting normalize=True and rounding the output to 3 decimal places. By default, value\_counts() method returns counts of unique values. By setting normalize=True, the value\_counts() will return the relative frequencies of the uniquevalues instead.**

**Task 6: Instructions**

**Split the Transfusion DataFrame into train and test datasets. Import train\_test\_split from sklearn.model\_selection module. Split Transfusion into X\_train, X\_test, y\_train and y\_test datasets, stratifying on the target column. Print the first 2 rows of the X\_train DataFrame with the head() method. Writing the code to split the data into the 4 datasets needed would require a lot of work. Instead, you will use the train\_test\_split() method in the scikit-learn library.**

**Task 7: Instructions**

**Use the TPOT library to find the best machine learning pipeline. Import TPOTClassifier from tpot and roc\_auc\_score from sklearn.metrics. Create an instance of TPOTClassifier and assign it to tpot variable. Print tpot\_auc\_score, rounding it to 4 decimal places. Print Id and transform in the for-loop to display the pipeline steps. You will adapt the classification example from the TPOT's Documentation In particular, you will specify scoring='roc\_auc' because this is the metric that you want to optimize for and add random\_state=42 for reproducibility You'll also use TPOT light Configuration with only fastmodels and preprocessors.The nice thing about TPOT is that it has the same API as scikit-learn, i.e., you first instantiate a model and then you train it, using the fit method.Data pre-processing affects the model's performance, and tpot's fitted\_pipeline\_attribute will allow you to see what pre-processing (if any) was done in the best pipeline.**

**Task 8: Instructions**

**Check the variance. Print X\_train's variance using var()method and round it to 3 decimal places.pandas.DataFrame.var() method returns column-wise variance of a DataFrame, which makes comparing the variance across the features in X\_train simple and straightforward.**

**Task 9: Instructions**

**Correct for high variance. Copy X\_train and X\_test into X\_train\_normed and X\_test\_normed Respectively. Assign the column name (a string) that has the highest variance to col\_to\_normalize variable. For X\_train and X\_testDataFrames:**