Data Analyst Project Proposal

Created by:

Ankit Hasija, MTE

Cover Letter

Hello,

I am pleased that you are being assigned the project of **Predict Blood Donations.** Review the following project proposal, which outlines the details of this project.

Thank you!
MTE!

Ankit Hasija

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 `tpot` library to automate your Machine Learning pipeline.
To complete this Project, you need to know some Python, pandas, and logistic regression.

Process

We will work closely with you to build and fulfill the needs of this project by the end of your internship. We will do this by establishing clear goals and a comprehensive solution based on project requirements.

Our process to achieve this is as follows:

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 columns.

Task 3: Instructions

Inspect the DataFrame's structure.

- Print a concise summary of the `transfusion` DataFrame with the `info()` method.

DataFrame's `info()` method prints some useful information about a DataFrame:

- index type
- column types
- non-null values
- memory usageincluding the index dtype and column dtypes, non-null values and memory usage.

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 `inplace` 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 unique values 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 `idx` 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 fast models 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_test` DataFrames:

    - Log normalize `col_to_normalize` to add it to the DataFrame.
    - Drop `col_to_normalize`.
- Print `X_train_normed` variance using `var()` method and round it to 3 decimal places.

`X_train` and `X_test` must have the same structure. To keep your code "DRY" (Don't Repeat Yourself), you are using a for-loop to apply the same set of transformations to each of the DataFrames.

Normally, you'll do pre-processing *before* you split the data (it could be one of the steps in machine learning pipeline). Here, you are testing various ideas with the goal to improve model performance, and therefore this approach is fine.

Task 10: Instructions

Train the logistic regression model.

- Import `linear_model` from `sklearn`.
- Create an instance of `linear_model.LogisticRegression` and assign it to `logreg` variable.
- Train `logreg` model using the `fit()` method.
- Print `logreg_auc_score`.

The `scikit-learn` library has a consistent API when it comes to fitting a model:

1. Create an instance of a model you want to train.
2. Train it on your `train` datasets using the `fit` method.

You may recognise this pattern from when you trained TPOT model. This is the beauty of the `scikit-learn` library: you can quickly try out different models with only a few code changes.

Task 11: Instructions

Sort your models based on their AUC score from highest to lowest.

- Import `itemgetter` from `operator` module.
- Sort the list of `(model_name, model_score)` pairs from highest to lowest using `reverse=True` parameter.

Congratulations, you've made it to the end!

Good luck and keep on learning!

If you are interested in learned what makes linear models so powerful and widely used, **Statistical Modeling in R** is a great resource! The coding is done in `R`, but it's the theoretical concepts that will help you to interpret the models you are building.

Dataset and Jupiter Notebook Files - Download from link below

https://drive.google.com/file/d/1S2o3wEAfEPha06ECh6kirwUijcCq54nY/view?usp=sharing

About MedTourEasy

MedTourEasy, a global healthcare company, provides you the informational resources needed to evaluate your global options. It helps you find the right healthcare solution based on specific health needs, affordable care while meeting the quality standards that you expect to have in healthcare. MedTourEasy improves access to healthcare for people everywhere. It is an easy to use platform and service that helps patients to get medical second opinions and to schedule affordable, high-quality medical treatment abroad.