Predicting Data Professionals' Salary with PyCaret

The goal of this project is to predict the salary of a data professional based on various characteristics using the PyCaret library. I used a publicly available dataset from Kaggle that contains information on over 23,997 respondents, including their job title, education level, years of experience, and salary.

Section 1 - Data Cleaning

The available data contains some missing observations, inconsistencies and irrelevant columns. The following steps were taken to prepare our data for modelling:

- The first row after the header contains questions from the survey and it is not relevant to our model. So, it was dropped from the dataset.

- Some columns in the dataset have lots of NaN values. Checking the number of unique values per column shows that some columns have only one unique value. These columns are zero variance predictors and do not add any meaningful information to our model. Columns with single value were selected and dropped from the dataset.

- The first column Duration (in seconds) shows the time taken to complete the survey in seconds and is not relevant to our model. Also, column information shows some columns have more than 80% missing values. These columns were dropped from the dataset.

- The salary column Q29 has 8136 non-null values and 15,861 missing values. We can fill the missing values with the mean, mode or constant. However, this is over 60% of total observations and may distort our prediction result. We opted for dropping the missing values to preserve originality in the dataset.

- Column headers in the dataset are not descriptive. We renamed column headers to indicate the values it contains.

- After getting and plotting the frequency of our target column salary, we discovered that there are too many categories. Also, the categories are inconsistent (some had '$' signs and others didn't). We removed the '$' sign from the categories and further grouped salary categories into buckets. The aim is to reduce the salary categories from twenty-six to fifteen or less.

Section 2 - Exploratory Data Analysis and Feature Selection

In order to demonstrate the *predict_model()* function on unseen data, a sample of 407 records has been withheld from the original dataset to be used for predictions. Another way to think about this is that these 407 records were not available at the time when the machine learning experiment was performed.

Setting the profile parameter to True displays a data profile for exploratory data analysis. Several pre-processing steps were performed in this experiment based on this analysis. Below are the steps taken in the setup:

- **Missing Values:** *academic_research* and *ml_experience* have missing values. Both columns are categorical and we used *imputation_type = 'simple'* and *categorical_imputation = 'constant'* to fill *"not_available"* for missing observations.

- **Multicollinearity:** there are high correlations between *coding_experience* and ml_experience, job_title and employer_industry which introduces multicollinearity into the data. We removed multicollinearity by using the remove_multicollinearity and parameter.

- **High Cardinality:** country has high cardinality with 58 unique values. We handled this using *high_cardinality_features* and *high_cardinality_method = 'frequency'* in the setup.

- **Fix Imbalance:** The target class salary is imbalanced with lots of observations in the first bucket. We fixed this using the *fix_imbalance*. When set to True, SMOTE (Synthetic Minority Over-sampling Technique) is used as a default method for resampling.

Before proceeding with the selection of models, we need to understand which metric is most valuable to us. Choosing a metric in a classification experiment depends on the problem we're trying to solve. There is always a trade-off between Precision and Recall. This means we have to choose and favour a balance between True Positives and False Negatives.

Here, the final model will be used to predict salary and accuracy is quite low. We want to choose a model with a better ability to have True Positives (better Recall), even if it comes with the cost of some False Positives (lower Precision). We need to however keep a tab on the trade-offs in Precision and AUC.

We will go ahead and create three models namely Random Forest Classifier (rf), Extra-Tree Classifier (et), Gradient Boosting Classifier (gbc) with best Recall and reasonable AUC/Precision.

Section 3 - Model Implementation and Tuning

Based on the model results we have, Gradient Boosting Classifier appears to be the best choice with the highest recall and AUC of 76%.

Hyperparameter Tuning

In order to find the best hyperparameters for the model, we used the *tune_model* function which loops through pre-defined hyper-parameter grids to find the best parameters for our model through 10 fold cross-validation. PyCaret uses the standard Randomized-Grid search to iterate through the parameters. The number of iterations *n_iter* was set to 50. Within the *tune_model* function, PyCaret also allows us to specify the metric we want to optimize. The default is Accuracy, but chose recall here since it is the metric we would want to increase/optimize.

Section 4 - Finalize Model

automl function in pycaret.classification and pycaret.regression will re-fit the model on the entire dataset. As such, for the model to be fitted on the entire dataset including the holdout set, *finalize_model* was explicitly used.

Section 5 - Predict on Unseen Data

We used *predict_model()* to generate prediction and the score (probability attached to prediction). The prediction dataframe also contains all the features and initial salary bucket for each observation. Out of the 5 columns shown in the prediction dataframe, our predicted one label correctly.