PRML Bonus Project

Title: Predict The Data Scientist's Salary In India

Debdut Saini (B20CS011)

Abstract:

The given dataset contains several parameters like the location of the job, experience required, skills required, company name, etc., and based on the data we need to predict the data scientist's salary.

Introduction:

Data Science is a growing field of statistics and provides an ample career opportunity. It is a highly demanding job in India. So, to make a correct decision in career choices, one might prefer to know the salary that a company might offer. So, it is worthy to make a machine learning model that roughly predicts the range of salary for the user.

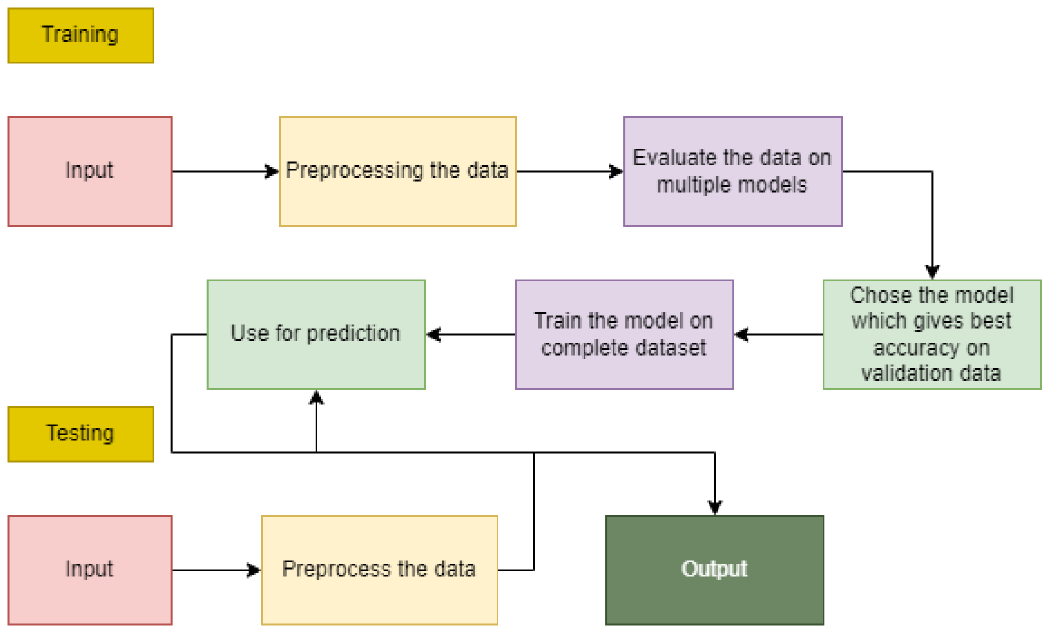
Dataset used: https://drive.google.com/drive/folders/1CZLJfivjgMPYtKOTepiYLbZ2iqvcVgsV

Overview:

The training dataset contains 8 columns out of which one is numeric and one is a label column, the rest 6 are object columns. Since the text columns contain most of the information, we can not encode it directly, rather we need to apply Natural Language Processing to extract the main information from the text data.
The original dataset contains 19802 rows.
Job_description column contains 4418 missing values, job_type contains 15005 missing values while the label contains 6 unique values.

Pipeline:

The pipeline takes the training data input and preprocesses to make it fit for training inbuilt models. The preprocessed data is then used for training the models. Try several combinations of models. Choose the model which gives the best performance on the validation set is chosen to evaluate the test data.


Training
Input
Preprocessing the data
Evaluate the data on multiple models
Chose the model which gives best accuracy on validation data
Train the model on complete dataset
Use for prediction
Testing
Input
Preprocess the data
Output

Preprocessing:

Preprocessing includes cleaning the data. It also includes lemmatization of the text to extract the important words.
Data is then converted to a numerical vector, where min_df is 0.1 and maximum is 0.9.
Note that: I have evaluated the models in cases:

- Case1: When there is no Laplace smoothing applied.
- Case2: When there is Laplace smoothing applied.

Models:

For each case, 5 models were trained and it was observed that for each case corresponding models perform almost similar. This is because we have chosen the min_df to be 0.1, if we chose it to be 0 then there will be a difference in accuracies.
Models used:

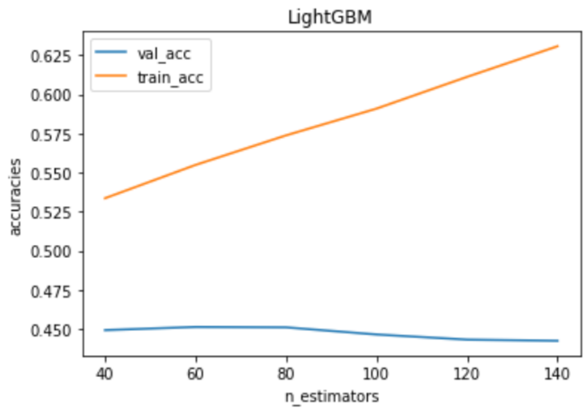
1. LightGBM with slight hyperparameter tuning. (Boosting)
2. SVM
3. RandomForest (bagging)
4. Multinomial Naive Bayes as it performs better for text classification
5. LDA as it separated two-class well considering the variance.

Evaluation Criteria Used:

- Overall Accuracy
- Since it is a classification problem with multiple classes, a classification report is used which gives precision and recall, and f1_score using the one vs all method.

Observations and Results:

It was observed that the lightGBM gave the best results on validation data. Therefore LightGBM is used to train over the whole data to make the final model which can be used to predict on any other test cases.

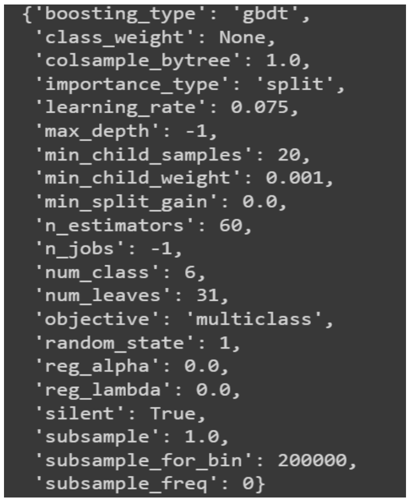


Light GBM accuracies with the different number of n_estimators.

Models	LightGBM	SVM	Random Forest	Multinomial NB	LDA
Overall validation accuracy	0.45140116132289826	0.23201211815198183	0.23201211815198183	0.25902549861146174	0.4034334763948498

As discussed above for both the cases with Laplace smoothing and without Laplace Smoothing the accuracies of corresponding models come out to be the same, this is because while converting the text data to numerical vector we set the min_df parameter to be 0.1, if we set it to 0 then we might observe different results.

The best parameters obtained are shown alongside.



```
{'boosting_type': 'gbdt',
 'class_weight': None,
 'colsample_bytree': 1.0,
 'importance_type': 'split',
 'learning_rate': 0.075,
 'max_depth': -1,
 'min_child_samples': 20,
 'min_child_weight': 0.001,
 'min_split_gain': 0.0,
 'n_estimators': 60,
 'n_jobs': -1,
 'num_class': 6,
 'num_leaves': 31,
 'objective': 'multiclass',
 'random_state': 1,
 'reg_alpha': 0.0,
 'reg_lambda': 0.0,
 'silent': True,
 'subsample': 1.0,
 'subsample_for_bin': 200000,
 'subsample_freq': 0}
```

Reference:

- CodeBasics
- NLP sources on Internet
- Regex documentation
- Sklearn documentation