1. **How long did it take you to solve the problem?**

Answer: It took me 7-8 hours to finish the assignment.

2. **What software language and libraries did you use to solve the problem? Why did you choose these languages/libraries?**

Answer: I have used “Python” as a programming language. I chose python because it provides excellent library support that helps implement ML tasks. To solve the assignment I used the following libraries

- Pandas: To store and transform the data
- Scikit-learn and numpy: For machine learning algorithms and math functions.
- Matplotlib and seaborn: To plot figures(used in Exploratory data analysis)
- Catboost: To train cat boost classifier.
- Lightgbm: To train lightgbm classifier
- Xgboost: To train xgboost classifier

3. **What steps did you take to prepare the data for the project? Was any cleaning necessary?**

Answer: followings are the steps and observations about the data

I. Read “train_features.csv” and “train_salaries.csv” and load them into the data frames. I have joined these 2 data frames into a single data frame using the “jobID” column.
II. I did an exploratory analysis of the data and found following
    A. There are 5 entries with 0 values for the “salary”. After inspecting further, I found that those jobs look not unpaid. Hence I decided to remove them.
    B. The salary distribution looks close to normal but there is a slight presence of the right trail.
    C. The dataset does not contain any missing values and it looks symmetric for the output and most of the features.
    D. All “jobIds” are unique so there is no need to remove duplicate data.

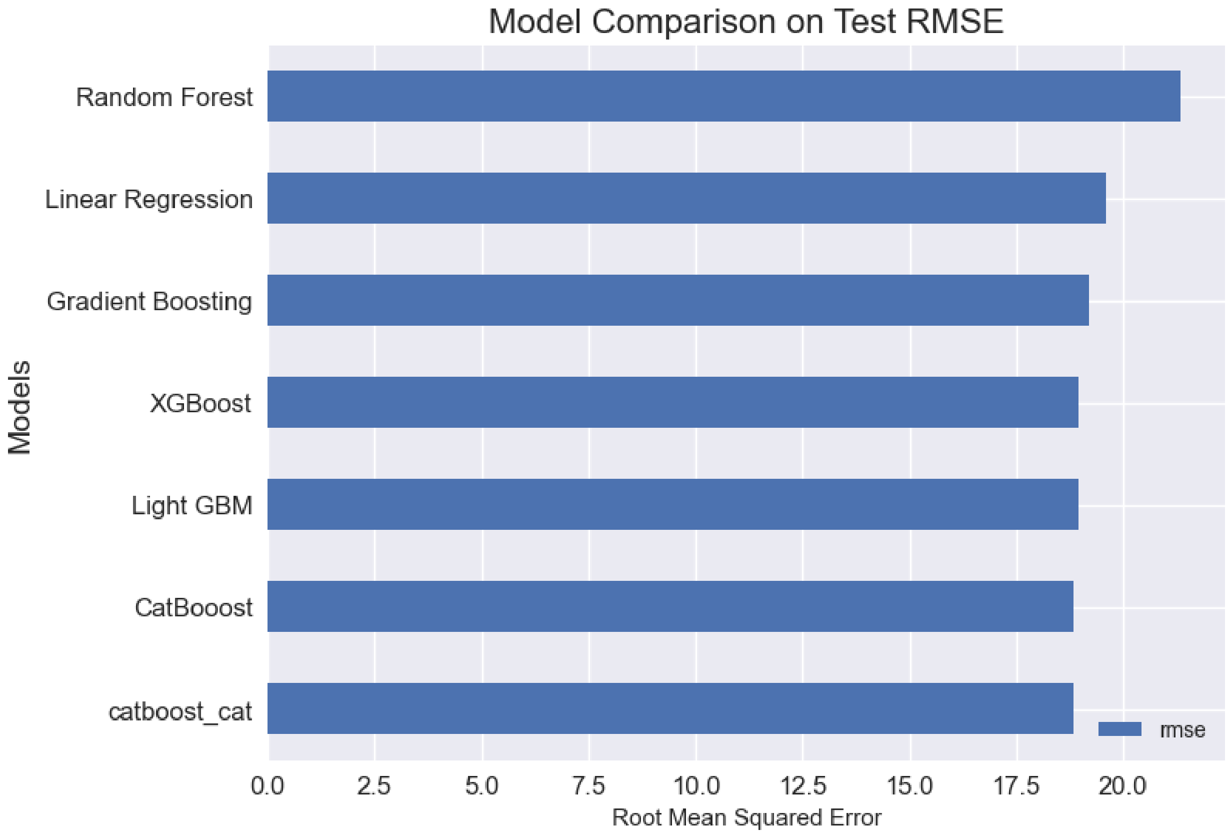
4. **About ML algorithms**

A. **What machine learning method did you apply?**
   Answer: I tried multiple ml models and choose one which worked best on the validation dataset. For this dataset, Catboost with categorical(non-encoded) variable worked best
B. **Why did you choose this method?**

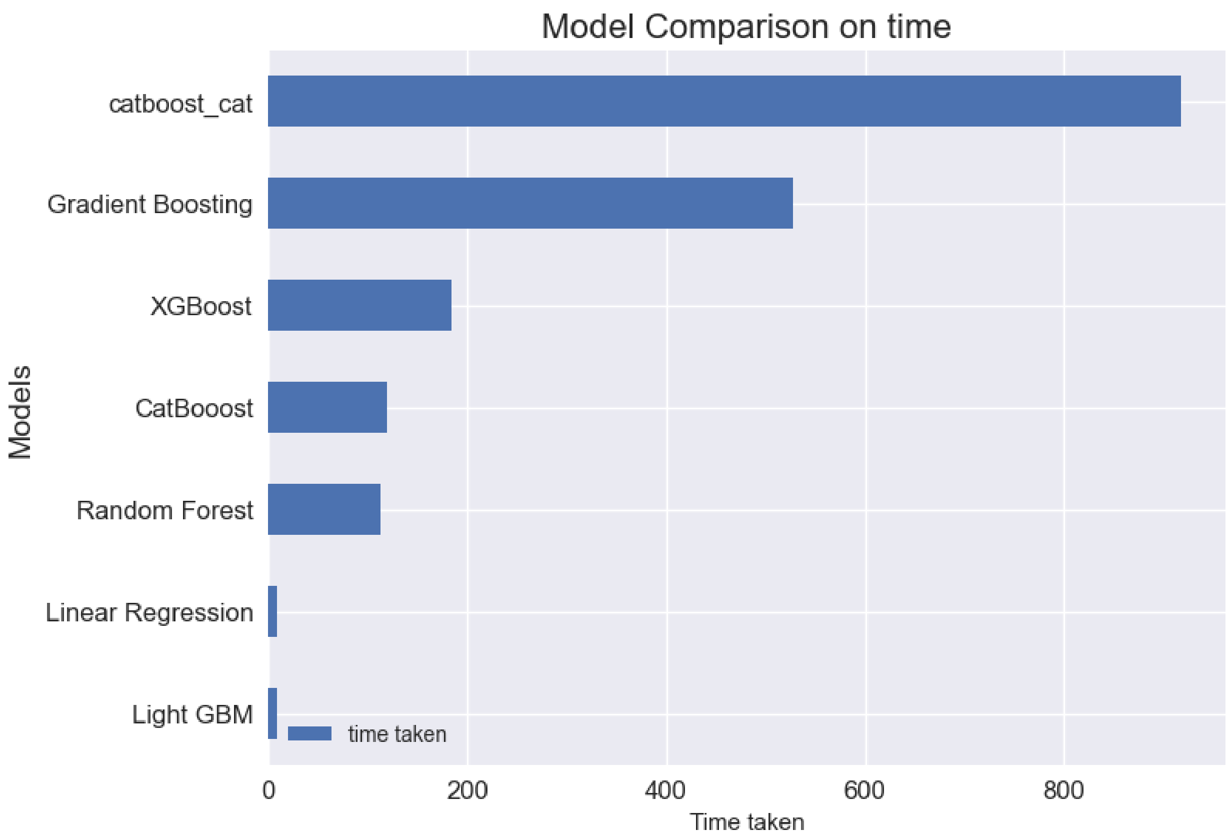
Answer: I choose Catboost because it works best on the validation dataset among all the tried algorithms. Catboost generally performs well when you have some categorical features in your feature set. In general boosting algorithms performs best when you have tabular data which included categorical and numerical features.

C. What other methods did you consider?

Answer: I tried many boosting models and validation on the validation dataset. Following is the list of models and their performances:



Catboost algorithm with non-encoded categorical features performed best with 18.84 RMSE scores. I also tried to fine-tune the Catboost algorithm with optuna hyperparameter tunning framework but RMSE error did not reduce much.



LightGBM and linear regression are the fastest among all the algorithms.

5. **Describe how the machine learning algorithm that you chose works.**

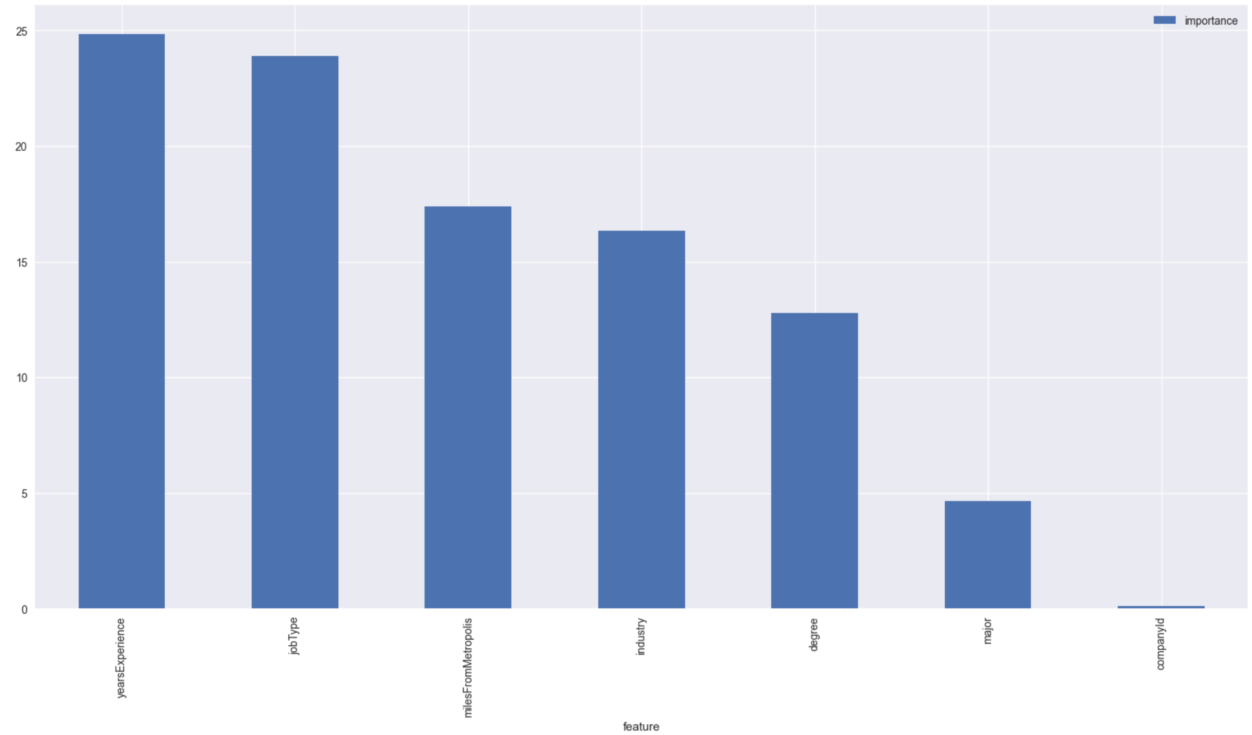
Answer: Catboost is the boosting algorithm designed to work very well with categorical features without any encoding. It works very well on Heterogeneous dataset i.e. dataset with many data types(categorical & numerical). CatBoost is the type of boosting algorithm. The main idea of boosting is to sequentially combine many weak models (a model performing slightly better than random chance) and thus through greedy search create a strong competitive predictive model. Because gradient boosting fits the decision trees sequentially, the fitted trees will learn from the mistakes of former trees and hence reduce the errors. This process of adding a new function to existing ones is continued until the selected loss function is no longer minimized. In the growing procedure of the decision trees, CatBoost does not follow similar gradient boosting models. Instead, CatBoost grows oblivious trees, which means that the trees are grown by imposing the rule that all nodes at the same level, test the same predictor with the same condition, and hence an index of a leaf can be calculated with bitwise operations. The oblivious tree procedure allows for a simple fitting scheme and efficiency on CPUs, while the tree structure operates as a regularization to find an optimal solution and avoid overfitting. Catboost is one of the fastest algorithm among all the boosting algorithm. Catboost uses the following parameters:

- Iterations: 1000 iterations means the CatBoost algorithm will run 1000 times to minimize the loss function.

- Loss Function: In here as we are classifying multiple classes we have to specify 'Multiclass'. In the case of Binary classification, it is okay if we don't mention the Loss Function the algorithm will understand and perform binary classification.
- bootstrap_type: This parameter affects the Regularization & speed of the algorithm aspects of choosing a split for a tree when building the tree structure. Here we have chosen Bayesian, But it is okay if we didn't specify this parameter.
- eval_metric: In here as we have to do multiclass classification we have chosen 'Multiclass' as eval_metric and when working with Binary Classification we don't have to specify this parameter.
- leaf estimation iterations: This parameter defines rules for calculating leaf values after selecting the tree structure, we have taken 100 but it is also okay to not specify this parameter.
- random strength: It specifies how random do we want our gradient boosting trees to be from each other. It is okay if we didn't specify.
- depth: How deep do we want our tree to be I have specified 7 because it gave me the highest accuracy but it is okay not to specify it and let the CatBoost algorithm use its default value.
- l2 leaf regularization: To specify the L2-regularization value, we have taken 5 but it's not mandatory.
- learning rate: It is very important but generally default CatBoost learning rate of 0.03 also works well.
- Bagging temperature: Defines the settings of the Bayesian bootstrap. It is used by default in classification and regression modes. Use the Bayesian bootstrap to assign random weights to objects. Not mandatory to specify.
- task type: It is very much recommended to use CatBoost algorithm with GPU only because with CPU CatBoost algorithm becomes quite slow.

6. **Was any encoding or transformation of features necessary? If so, what encoding/transformation did you use?**
Answer: For catboost with categorical feature, I did not use any encoding because it was not needed. But for algorithms like linear regression and other boosting algorithm(Gradient Boosting) encoding of categorical features are necessary. I tried One hot encoding of categorical features and log & square root transformation of numerical features to introduce non linearity. I also did scaling of numerical features using "MinMaxScaler" from sklearn library.

7. **Which features had the greatest impact on salary? How did you identify these to be most significant? Which features had the least impact on salary? How did you identify these?**
Answer: Below is the feature importance plot:



As given in above figure “yearsExperience” had the greatest impact on the salary. “companyId” and “major” are the least impactful features. I used trained boosting model “feature_importances_” attribute to get this. For all the boosting model we can calculate feature importance easily because it’s inbuilt.

8. **How did you train your model? During training, what issues concerned you?**

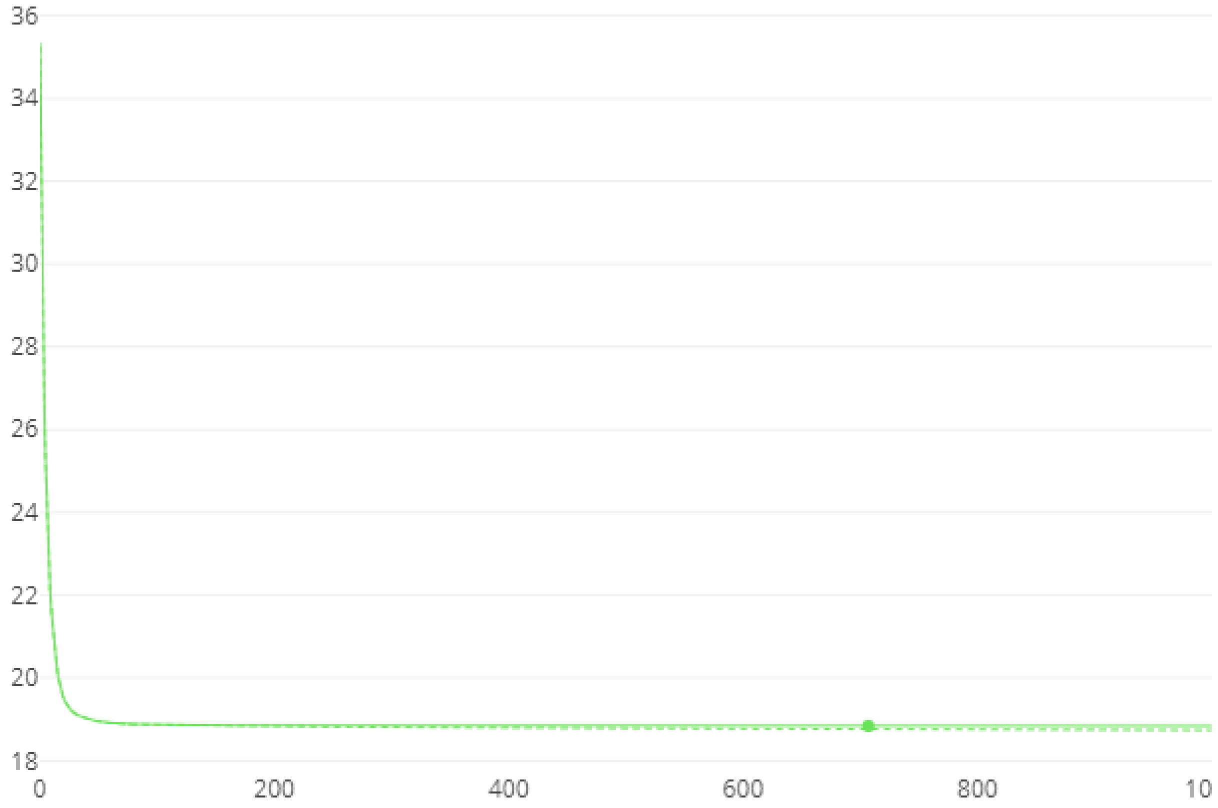
Answer: I have divided the given dataset(train_features.csv) into 2 parts: train(80%) and validation(20%) set. I trained my model on the train set and calculate estimates on validation set. I have compared all the algorithms on the same validation set. For training I used “CatBoostRegressor” function from “catboost” library. Below is the code for training catboost with categorical features:

```
catb_cat = CatBoostRegressor(loss_function='RMSE',
                             random_state=1,
                             verbose=False,
                             task_type='CPU')

pool_train = Pool(non_encoded_X_train,
                  non_encoded_Y_train,
                  cat_features = ['industry', 'companyId', 'jobType',
'degree', 'major'])

pool_val = Pool(non_encoded_X_val,
                cat_features = ['industry', 'companyId', 'jobType',
'degree', 'major'])

# eval_dataset = Pool(non_encoded_X_val, non_encoded_y_val)

start = time.time()
catb_cat.fit(pool_train,
                  eval_set=(non_encoded_X_val, non_encoded_Y_val),
                  early_stopping_rounds=791, plot=True)
```

Issues: I got memory error because of the dataframe size. To solve this i have changed the datatypes of categorical features from “object” to “category”. By default panda treat string column as objects. Below is the code for that:

```
#https://www.dataquest.io/blog/pandas-big-data/
for col in ["companyId","jobType","degree","major","industry"]:
    train_feat_sal[col] = train_feat_sal[col].astype('category')
```

I also tried doing hyperparameter tuning using optuna framework. Optuna is library for efficient tunning hyperparameters. It took lot of time(almost 4-5 hours) and results didn’t improve much.



Above is RMSE plot(y-axis) with no of iterations on X-axis

9. **a) Please estimate the RMSE that your model will achieve on the test dataset.**
Answer: Below are the estimates for different models:

ML Algorith	Validation set RMSE
Linear Regression	19.61
Random Forest	21.33
Gradient Boosting	19.20
Light GBM	18.93
XGBoost	18.96
Catboost with encoded cat features	18.85

Catboost with non encoded cat features	18.84

As shown in the table, Catboost with non encoded cat features performed best with RMSE score 18.84.

b) **How did you create this estimate?**
Answer: To create this estimate, i have divided the given dataset(train_features.csv) into 2 parts: train(80%) and validation(20%) set. I trained my model on the train set and calculate estimates on validation set. I have compared all the algorithms on the same validation set.

10) **What metrics, other than RMSE, would be useful for assessing the accuracy of salary estimates? Why?**
Answer: We can also use the following alternative metrics.
R-squared: R-square tells the proportion of variance in the dependent variable(salary) that can be explained by the independent variable(all the features).

Huber loss: The problem with RMSE is that it is not robust to outliers. Huber loss is less sensitive to outliers compared to RMSE. Huber loss combines the advantages of Squared loss and Absolute loss. More details are given here:
https://towardsdatascience.com/regression-in-the-face-of-messy-outliers-try-huber-regressor-3a54ddc12516

Future work: The model can be improved by trying more transformation of features like Target based encoding.

Note: Most of the figures related to EDA are there in the code(Assignment.ipynb)