1. **How long did it take you to solve the problem?**
Answer: I gave around 3 days to solve/optimize the assignment (with 2-3 hours of sitting per day) given to me

2. **What software language and libraries did you use to solve the problem? Why did you choose these languages/libraries?**
Answer: I used python to solve this task. Here is the list of libraries I have chosen to accomplish this task

- pandas - To do analysis on the dataset.
- plotly - To plot how salary is varying with different columns in the dataset.
- sklearn - To see how well this dataset is working with various linear regression models
- xgboost - An ensemble learning method to see how well is performing on the dataset.
- lightgbm - A gradient boosting framework to see how well is performing on the dataset.
- keras - used for building neural networks to work with the non linearity in the dataset.
- matplotlib - To plot graphs how the loss function is behaving for every epoc.

3. What steps did you take to prepare the data for the project? Was any cleaning necessary?
Answer: Steps taken to prepare data for the assignment-

- Merge the train_features and train_salaries data frame on the common column of job-id
- Check if any of the columns in the above data frame contains null values or not.
- Plot the distribution of salary with the column containing categorical features like
    - jobType
    - degree
    - major
    - industry
    - companyId
- Used the above plots to ignore the features having least impact on the salary
- Added extra column 'Type' with all values set to 'Train' on the above data frame
- Added extra column 'Type' with all values set to 'Test' on the above test data frame
- Added extra column 'salary' with all values set to '0' on the above data frame
- Combine the 2 data frames together so that the same cleaning steps can be applied on the test and train data frame.

4. a) What machine learning method did you apply?
Anwer: I used keras sequential model to add multiple neural network layers with 'relu' activation function to solve this problem.
b) Why did you choose this method?
Answer: I earlier tried various Linear Regression and Random forest approaches to solve this problem, but they were not able to get the desired results because linear models fail to capture the non-linearity in the dataset.

c) What other methods did you consider?
Answer: I tried the following methods before jumping to use neural network to solve this problem

- Ridge Regression (sklearn linear model)
- SGDRegressor (sklearn linear model)
- ElasticNet (sklearn linear model)
- RandomForestRegressor (sklearn ensemble method)
- GradientBoostingRegressor (sklearn ensemble method)
- XGBRegressor (from xgboost)
- LGBMRegressor (from lightgbm)

5. Describe how the machine learning algorithm that you chose works.
Anwer: I am using neural network architecture to solve this problem. I have added multiple layers (with varying sizes).Every layer in the network except the last layer is followed by an activation function (to capture the non-linearity).

To help the network generalize better and converge faster I have used Batch Normalization and Dropout layers to the internal layers

The input to the 1st layer of the network is the dataset itself and the input to the subsequent layers is the output of the previous layers. Training the network is a 2 step process, one we call **feed-forward** step and the other we call **back-propagation.**

In **Forward propagation** the input data is fed in the forward direction through the network. Each hidden layer accepts the input data, processes it as per the activation function and passes to the successive layer. **Back Propagation** helps to calculate the gradient of a loss function with respect to all the weights in the network.

We do this for multiple epochs/steps until the network starts to overfit the data

6. Was any encoding or transformation of features necessary? If so, what encoding/transformation did you use?
Answer: There are some categorical features in the dataset for example - 'jobType', 'degree', 'major' and 'industry'. I used pandas get_dummies function on the above features to convert categorical variable into one hot encoded vectors

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:

- Features with the **greatest impact** on the salary - **milesFromMetropolis** & **yearsOfExperince**.
- Features with the **least impact** on the salary - **companyId**.

I have identified these by plotting histograms for these features against the salary values.

8. How did you train your model? During training, what issues concerned you?
Answer: I used keras **sequential model.fit** to train my model. I used root mean square error as my loss function and trained the model for 10 epocs and used 'adam' optimizer to solve the problem.
During training I have to come up with my own custom function to calculate the root mean squared error. To achieve the same I used the following code.

```
from keras import backend
def rmse(y_true, y_pred):
  return backend.sqrt(metrics.mean_squared_error(y_true, y_pred))
```

Once the training is complete, I tried loading the model using the **load_model** function from keras. But it failed to load the model. The reason is the use of a custom loss function defined for training the model. To fix this I have to come up with the below code to load the model.

```
from keras.models import load_model
model = load_model('salary_predictor.h5', custom_objects={'rmse': rmse})
```

9. a) Please estimate the RMSE that your model will achieve on the test dataset.
Answer: Expected RSME on the test dataset would be around 16%.
b) How did you create this estimate?
Answer: I reserved 10% of the training data for validation and I am getting around the same RMSE error on the validation set.

10. What metrics, other than RMSE, would be useful for assessing the accuracy of salary estimates? Why?
Answer: I have chosen RMSE to train the model. Apart from this we can also use **R-squared** error and **Mean absolute error**.

1. R-squared is a measure of how close the data are to the fitted regression line. It is also known as the **coefficient of determination**, It is the percentage of the variation that is explained by a **linear model**. In general, **the higher the R-squared, the better the model fits the data.**

R-squared = Explained variation / Total variation. lies between 0 and 100%:

- 0% indicates that the model explains none of the variability of the data around its mean.
- 100% indicates that the model explains all the variability of the data around its mean.

2. Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.