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

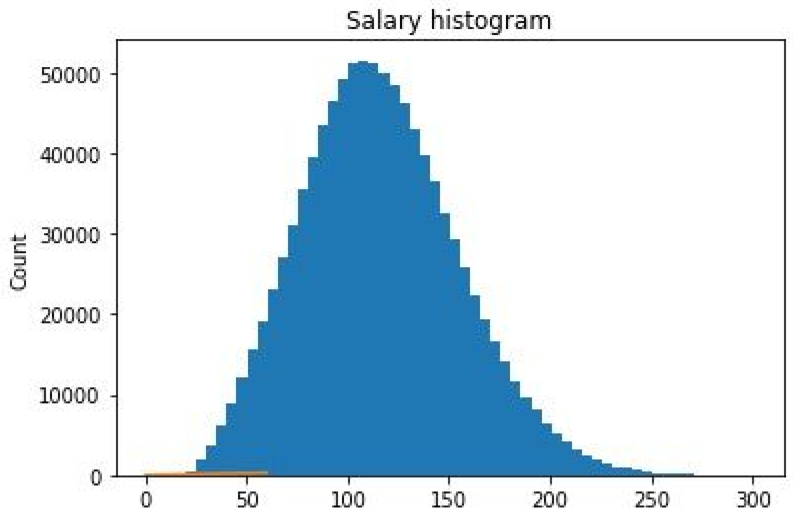
2. What software language and libraries did you use to solve the problem?

- Software language: Python
- Libraries: Graphlab Create, pandas, numpy, sklearn, matplotlib, pandas_profiling, xgboost

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

I used graphlab create to prepare the data for the project. I took the following steps:

1. Summary of data types: 7 categorical columns and 2 numerical
2. Noticed that the histogram of the target column, ‘salary’, is right-skewed. There is a right tail of values toward larger salaries.
3. By observing the distribution of salaries for different combinations of categorical and numerical data, noticed that all of them have the same underlying shape, a positively skewed distribution.
4. We would have to keep an eye on the residuals and find the model that can give us a gaussian distribution on them.
5. The models that were the best behaved for the residuals were: Boosted Decision Trees and XGBoost regression.
6. I used the residuals from the first training of an XGBoost model and removed the data points 2.5 sigma away from zero.
7. Then I used the remaining data to train a new XGBoost model that became the final model for the predictions.



4. What algorithmic method did you apply? Why? What other methods did you consider?

I tried regression using the following regression models:

1. Linear Regression
2. Boosted Decision Trees
3. Random Forest
4. Extreme Gradient Boosting
5. Finally, I used the results from Extreme Gradient Boosting, to clean the data.
6. Then trained an Extreme Gradient Boosting regression model on the clean data to finalize the predictor.

5. Describe how the algorithmic method that you chose works?

I used Graphlab create for linear regression, Boosted decision trees and Random Forest.

1. Linear Regression: Graphlab-create basically dummy encodes all categorical data and then optimizes the weights for each feature that would allow us to predict the best predict Salary or minimize the error. The algorithm uses Newton solver to achieve the minimal distance between predicted “salary” for train data.
2. Gradient Boosted Decision Trees: It builds a set of tree like object, with a certain depth, and a number of nodes called weak learners, and then it combines the results to make a strong learner. The trees are build on a stage-like fashion and it on the residuals at each stage. The data get split at each node according to the predicting power of the split or minimizing sum of squared errors. The splitting is done recursively until you reach the final nodes that give you an average value of the quantity of interest. The predicted value of the first tree can be added to the

original dataset as a new feature and then you allow the algorithm build a second trees that includes the output of your first Tree as an input. This technique is referred as boosting trees.

3. Random forest: It also builds trees with a certain depth and a number of nodes. For this case, each tree operates independently and it concentrates on a random subset of features at a time. For the regression model, you then average the predictions coming from each tree. Very useful to prevent overfitting.
4. Extreme Gradient Boosting, similar to Gradient Boosted Decision trees but with regularization included, to prevent overfitting.

6. What features did you use? Why?
All features were included. The categorical features such as: `industry, jobType, degree, major` and `companyId` were dummy encoded. For the regressions to work, you need numerical values that can be easily split at nodes, or weighted by coefficients.

In all regressions attempted, `companyId` finished as the least important feature. For linear regressions it has the smallest "coefficients" and for Boosted Decision Trees and Random Forest regressions, it was the feature with the least amount of node counts. We can consider not including it in the regression, if reduction of features is needed. For our sample size and computing power, it was possible to include them and train our models.

7. How did you train your model? During training, what issues concerned you?
I trained my models with 80% of the data available to train, and kept 20% of the data for evaluating the performance. One can also do cross validation, and split the data into 5 fold samples to measure performance by training on 5 different 80% samples and evaluating the performance on the 20% remaining. This would help us gain confidence in the stability of our model and the hyper-parameters chosen.

The main issues that I was concerned about were:

- Will the size and sparsity of the matrix (from the dummy encoding) prevent the regression models from converging?
- Is the skewness of the salary distribution captured by the features given?
- I am concerned about the residuals measured, they are not gaussian. This is certainly related to the lack of power the features have to reproduce the skewness of our dependent variable, salary. Do we need to transform the dependent variable? Take the log or the square root? Do we need to transform

any of the numerical features: yearsExperience or milesFromMetropolis and see if we can improve the prediction power of our modeling?

8. How did you assess the accuracy of your predictions? Why did you choose that method? Would you consider any alternative approaches for assessing accuracy?

There were several metrics that were helpful to evaluate the accuracy of the predictions: Root mean square error (RMSE), max error and took a look at the distribution of the residuals. The final RMSE is 18.9 and the maximum error experienced was 186.87. There seem to be a small percentage of the entries that seem really hard to predict. We can remove them with a threshold and run the models again, in an attempt to keep entries that can be better suited for a prediction with our models.

9. 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?

According to the list of important features in our Boosted Decision Tree model, the top ones are (and the number of times they appear in a node):

- milesFromMetropolis - 437 times
- yearsExperience - 434
- industry - Education - 123

the ones with the least impact were with very few nodes using them to split the data:

- All companyIds

The library from graphlab-create, allows you to print the summary of important features ordered by the number of times they were used to split the data in a node.