Capstone Project

Machine Learning
Engineer Nanodegree

Ninh Sai
Feb 3rd, 2018

Job Salary Prediction

Definition

Project Overview

Job seekers find jobs always want to know salary offer while there are still many employers don't want to publish salary of posting jobs. Vietnamworks has only 36% jobs with visible salary. Adzuna stated 5 years ago that approximately half of the UK job ads they index have a salary publicly displayed. I'd like to improve job seekers' experience by providing a predicted salary of any jobs based on some public information of jobs. Predict salary for a user profile/resume will be similar and can use this model to have good prediction engine. This will be a useful tool for job seekers too.

I'd like to do job salary prediction based on public information of jobs like Job Title, Job Level, Location, Industry, Company, Company Size, Years of Experience, Skill and Job Description to provide this useful information to job seekers. These all fields are important.

To have a legal good dataset in short time I use data set of Job Salary Prediction competition on Kaggle. Adzuna collected job ads from over 100 UK sources and normalized some fields. This dataset doesn't have all above fields but have some important fields and have full description field.

Problem Statement

The goal is to create an engine to predict job salary based on job's public information (a regression supervised learning problem). Giving a job or a list of jobs, with information about job title, job description, location, company name, contract type, contract time, the engine can predict salary with a good accuracy.

Tasks:

- Download Kaggle's Job Salary Prediction dataset and analyse to a good overview about the dataset and understand each of fields.

- Build an engine can do preprocessing data set (vectorize, transform), train then predict on unseen data.
- Try 3 algorithms, GBM, XGBoost and lightGBM, with grid search on different parameters to find the best fit algorithm and its parameters for this problem.

Metrics

Mean absolute error (MAE)

MAE is a measure of the difference between two continuous variables so it is a good metric to know how accurate the engine is.

Good rate

Because MAE is the different value, a small error number is still bad if the real value is equal or even smaller than the error number. For example, 5,000 error is good if the real value over 200,000 (the highest salary in the dataset) but very bad if the real value is 5,000 (the smallest salary).

I calculate the **Good rate** to see how many percentages of jobs the engine can predict with acceptable differences. The higher Good rate the better engine.

The acceptable difference has a percent variance equal or smaller than 30%.

I set a prediction is good if percent variance is less than or equal 30%. If real salary is 1,000, good predicted range is [700-1,300]. If predicted value is smaller than 700 or greater than 1,300 it is a not good prediction.

$$\text{Good rate} = (N_{\text{PercentVariance} <= \text{Threshold}} / N_{\text{total}})$$

$N_{\text{diff\%} < \text{Threshold}}$: Number of jobs having percent variance <= Threshold

The Threshold is 30% (smaller is better). Predicted value should be in the range of [real - 30%, real+ 30%].

Analysis

Data Exploration

Download the dataset

I use Train_rev1.gzip from:
https://www.kaggle.com/c/job-salary-prediction/data

I copy their data description here for fast reference:

The main dataset consists of a large number of rows representing individual job ads, and a series of fields about each job ad. These fields are as follows:

- Id - A unique identifier for each job ad
- Title - A free text field supplied to us by the job advertiser as the Title of the job ad. Normally this is a summary of the job title or role.
- FullDescription - The full text of the job posting as provided by the job advertiser. Where you see ***s, we have stripped values from the description in order to ensure that no salary information appears within the descriptions. There may be some collateral damage here where we have also removed other numerics.
- LocationRaw - The free text location as provided by the job advertiser.
- LocationNormalized - Adzuna's normalized location from within our own location tree, interpreted by us based on the raw location. Our normalizer is not perfect!
- ContractType - full_time or part_time, interpreted by Adzuna from the description or a specific additional field we received from the advertiser.
- ContractTime - permanent or contract, interpreted by Adzuna from the description or a specific additional field we received from the advertiser.
- Company - the name of the employer as supplied to us by the job advertiser.
- Category - which of 30 standard job categories this ad fits into, inferred in a very messy way based on the source the ad came from. We know there is a lot of noise and error in this field.
- SalaryRaw - the free text salary field we received in the job advert from the advertiser.
- SalaryNormalised - the annualized salary interpreted by Adzuna from the raw salary. Note that this is always a single value based on the midpoint of any range found in the raw salary. This is the value we are trying to predict.
- SourceName - the name of the website or advertiser from whom we received the job advert.

All of the data is real, live data used in job ads so is clearly subject to lots of real-world noise, including but not limited to: ads that are not UK based, salaries that are incorrectly stated, fields that are incorrectly normalized and duplicate adverts.

Some data samples

I show below only 2 samples with small image size because there is a rather big field, FullDescription. Please zoom in to see clearer.

Id	Title	FullDescription	LocationRaw	LocationNormalized	ContractType	ContractTime	Company	Category	SalaryRaw	SalaryNormalized	SourceName
12612628.0	Engineering Systems Analyst	Engineering Systems Analyst Dorking Surrey Salary ****K Our client is located in Dorking, Surrey and are looking for Engineering Systems Analyst our client provides specialist software development Keywords Mathematical Modelling, Risk Analysis, System Modelling, Optimisation, MISER, PIONEEER Engineering Systems Analyst Dorking Surrey Salary ****K	Dorking, Surrey, Surrey	Dorking	?	permanent	Gregory Martin International	Engineering Jobs	20000 - 30000/annum 20-30K	25000.0	cv-library.co.uk
12612830.0	Stress Engineer Glasgow	Stress Engineer Glasgow Salary **** to **** We re currently looking for talented engineers to join our growing Glasgow team at a variety of levels. The roles are ideally suited to high calibre engineering graduates with any level of appropriate experience, so that we can give you the opportunity to use your technical skills to provide high quality input to our aerospace projects, spanning both aerostructures and aeroengines. In return, you can expect good career opportunities and the chance for advancement and personal and professional development, support while you gain Chartership and some opportunities to possibly travel or work in other offices, in or outside of the UK. The Requirements You will need to have a good engineering degree that includes structural analysis (such as aeronautical, mechanical, automotive, civil) with some experience in a professional engineering environment relevant to (but not limited to) the aerospace sector. You will need to demonstrate experience in at least one or more of the following areas: Structural/stress analysis Composite stress analysis (any industry) Linear and nonlinear finite element analysis Fatigue and damage tolerance Structural dynamics Thermal analysis Aerostructures experience You will also be expected to demonstrate the following qualities: A strong desire to progress quickly to a position of leadership Professional approach Strong communication skills, written and verbal Commercial awareness Team working, being comfortable working in international teams and self managing PLEASE NOTE SECURITY CLEARANCE IS REQUIRED FOR THIS ROLE Stress Engineer Glasgow Salary **** to ****	Glasgow, Scotland, Scotland	Glasgow	?	permanent	Gregory Martin International	Engineering Jobs	25000 - 35000/annum 25-35K	30000.0	cv-library.co.uk

“?”: missing value.

Exploratory Visualization

H2O is excellent at the user interface. I use its web-based interactive environment, [H2O Flow](), to analyze the dataset.

Default link to local H2O Flow: http://localhost:54321/flow/index.html

The dataset has 244,768 jobs, 12 columns with 412MB compressed size.

The dataset is highly skewed to the left, 75% of jobs have the lower salary than 25% of max salary. There are 3 fields having high missing rate: ContractType, ContractTime and Company. All other fields have almost no missing.

With domain knowledge, I see all provided fields are necessary to predict salary. There are 2 raw fields (LocationRaw and SalaryRaw) I will ignore in this capstone because their normalized fields look good and I don't have much time to analyze raw fields.

Train_rev1.hex

Actions: View Data | Split... | Build Model... | Predict | Download | Export

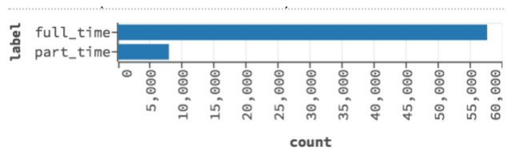
Rows	Columns	Compressed Size
244768	12	412MB

COLUMN SUMMARIES

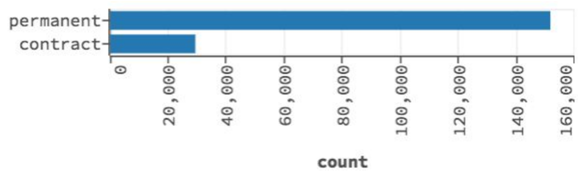
label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality
Id	int	0	0	0	0	12612628.0	72705235.0	69701420.7988	3129813.3458	·
Title	string	1	0	0	0	·	·	·	·	·
FullDescription	string	0	0	0	0	·	·	·	·	·
LocationRaw	enum	0	92	0	0	0	20985.0	·	·	20986
LocationNormalized	enum	0	1	0	0	0	2731.0	·	·	2732
ContractType	enum	179326	57538	0	0	0	1.0	0.1208	0.3259	2
ContractTime	enum	63905	29342	0	0	0	1.0	0.8378	0.3687	2
Company	enum	32430	1	0	0	0	20811.0	·	·	20812
Category	enum	0	21846	0	0	0	28.0	·	·	29
SalaryRaw	enum	0	1	0	0	0	97276.0	·	·	97277
SalaryNormalized	int	0	0	0	0	5000.0	200000.0	34122.5776	17640.5431	·
SourceName	enum	1	58	0	0	0	166.0	·	·	167

- Title and FullDescription are strings and FullDescription has long strings causing big size for the dataset. Only 1 jobs missed Title. These fields are important to use.

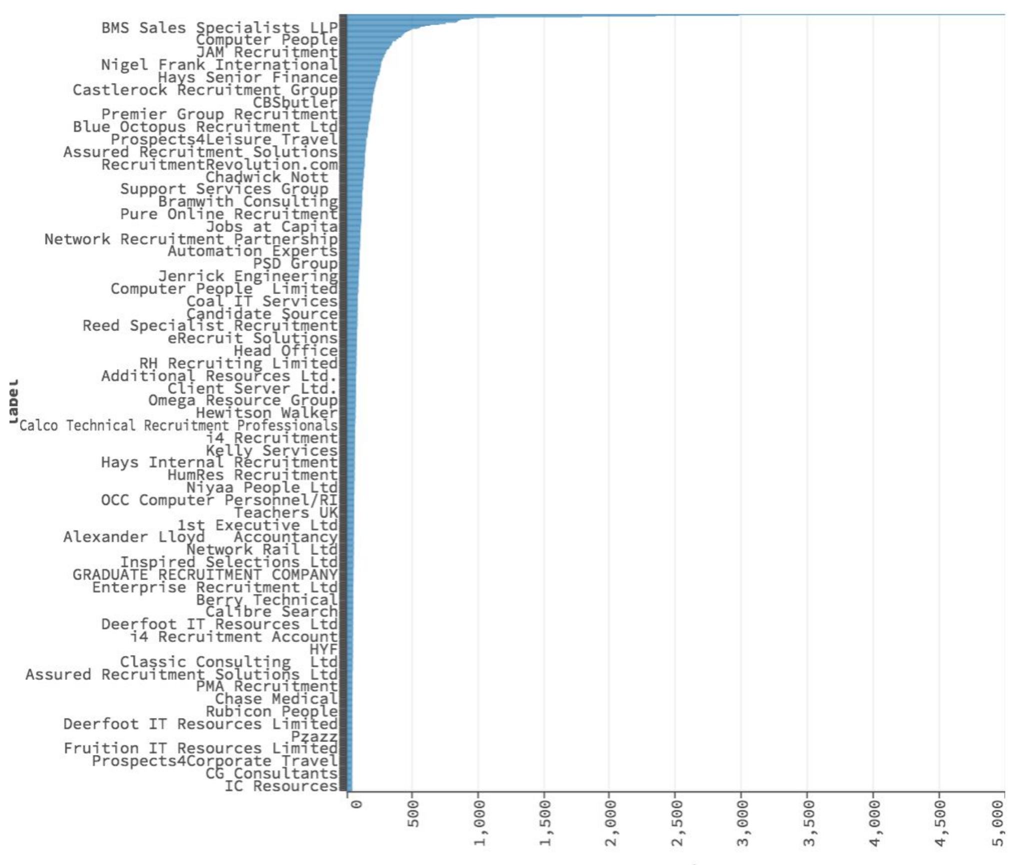
- LocationRaw has 20,986 unique values but Adzuna normalized to only 2732 LocationNormalized unique values. I will use LocationNormalized and drop LocationRaw (even though they said their normalizer is not perfect, I still think it's much better than their raw values when having more time I can analyze LocationRaw to improve)
- ContractType: There are 2 distinct kinds, 'full_time' and 'part_time'. More than 73% rows missed value.



- ContractTime: there are 2 unique types, 'permanent' and 'contract', 26% are missing.

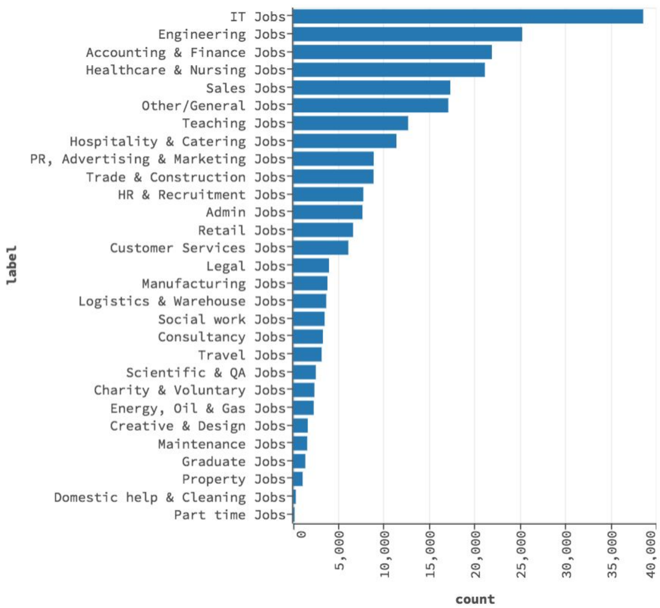


- Company: There are 20,812 companies. 13% jobs have no company information.



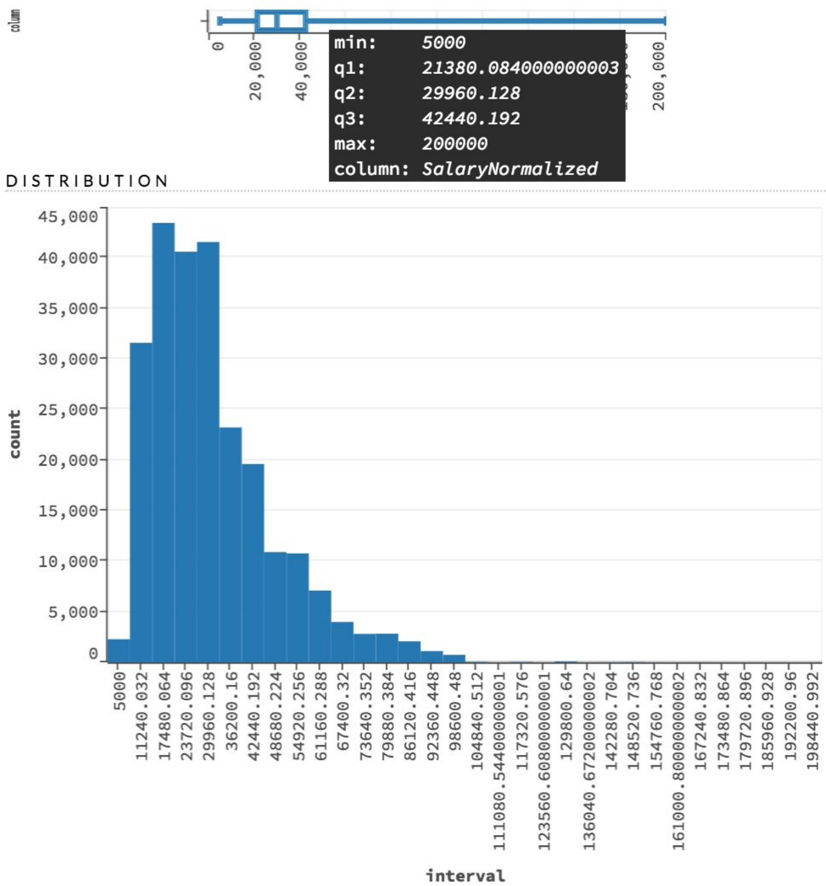
- Category: 29 categories. No missing.

Top 3 categories are: “IT”, “Engineering” and “Accounting & Finance”. Their jobs occupy 35% total jobs (85,503/244,768).



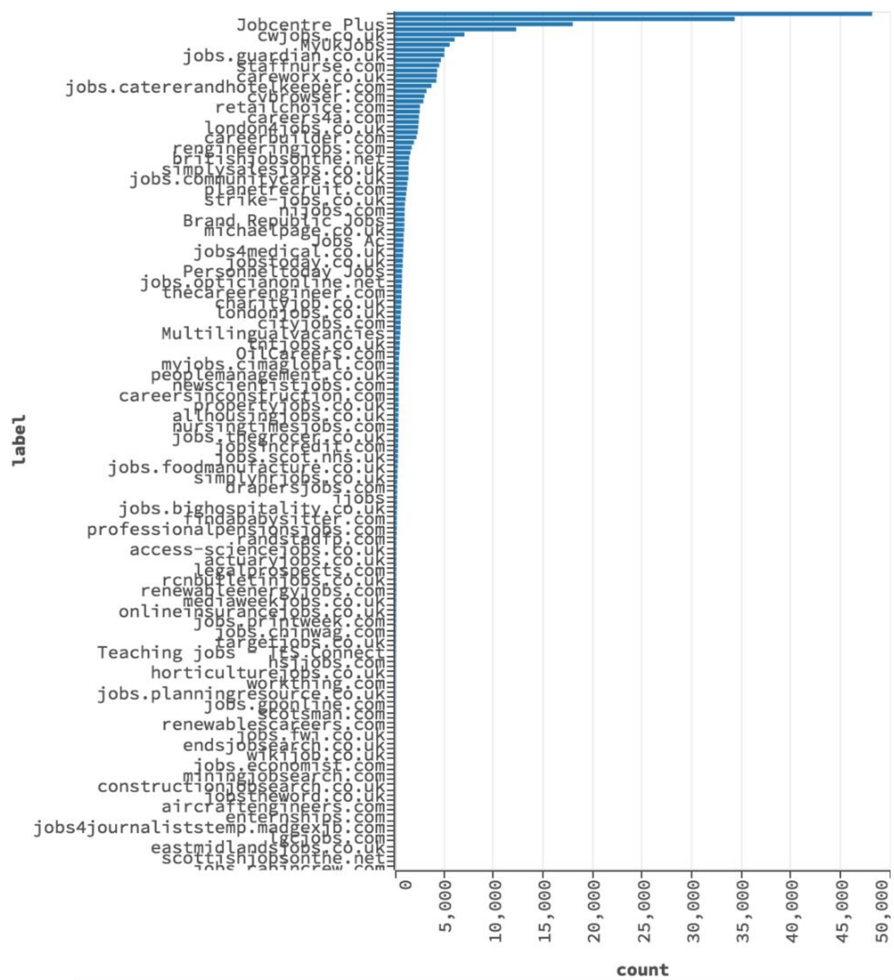
- SalaryRaw is text so ignore this time.
- SalaryNormalized: min 5,000, max 200,000 and mean about 34,000.

The below distribution chart shows SalaryNormalized is highly right skewed. 75%(q3) jobs have under 43,000 (which is smaller than 25% of the max SalaryNormalized).



In above histogram, we can see 98,600 is the biggest salary we can see its blue column with 735 data samples.

- SourceName: There 167 unique SourceName values. Only 1 missing case.



"IT jobs" is the 1st category with the highest number of jobs, I can use only IT jobs if I want to dig deeper but run all whole dataset is slow. Below is the summary statistic about it in the dataset.

topCatsJobs_df

Actions: View Data | Split... | Build Model... | Predict | Download | Export

Rows	Columns	Compressed Size
38483	10	69MB

COLUMN SUMMARIES

label	type	Missing	Zeros	+Inf	-Inf	min	max	mean	sigma	cardinality	Actions
Id	int	0	0	0	0	27527047.0	72695949.0	69858811.8156	2455258.4953	·	Convert to enum
Title	string	0	0	0	0	·	·	·	·	·	Convert to enum
FullDescription	string	0	0	0	0	·	·	·	·	·	Convert to enum
LocationNormalized	enum	0	0	0	0	2.0	2730.0	·	·	2732	Convert to numeric
ContractType	enum	33600	4830	0	0	0	1.0	0.0109	0.1036	2	Convert to numeric
ContractTime	enum	2376	3749	0	0	0	1.0	0.8962	0.3050	2	Convert to numeric
Company	enum	8379	0	0	0	5.0	20810.0	·	·	20812	Convert to numeric
Category	enum	0	0	0	0	13.0	13.0	·	·	29	Convert to numeric
SalaryNormalized	int	0	0	0	0	5000.0	190809.0	43983.9114	18241.4438	·	Convert to enum
SourceName	string	0	0	0	0	·	·	·	·	·	Convert to enum

Algorithms and Techniques

Salary prediction is a regression supervised learning problem.

Given a rather big dataset (more than 244 thousands of rows in over 400 MB) of job ads with 12 fields in 3 main data types (string, int, and enum). Target field contains continuous numbers.

As my analysis in Domain Background and Datasets parts of my proposal, all fields provided are related to salary. 2 raw fields are ignored in this capstone. Some useful fields are missed like Job Level, Years of Experience, Benefit, etc. But these missed fields can be included in Title and FullDescription. FullDescription is a big string field with long text. The dataset is highly skewed to the left, 75% of jobs have the lower salary than 25% of max salary.

This kind of problem can be solved by deep neural networks (solution of the first prize), random forest (popularly shared). I found GBM is better fit for this dataset.

Title and FullDescription are text so I'll vectorize them using H2O's Word2vec to have good vectors representing well the important meaning of job title and full description in the way easy to feed into a machine to learn.

Summary, the below table shows GBM is a good try for Salary Prediction:

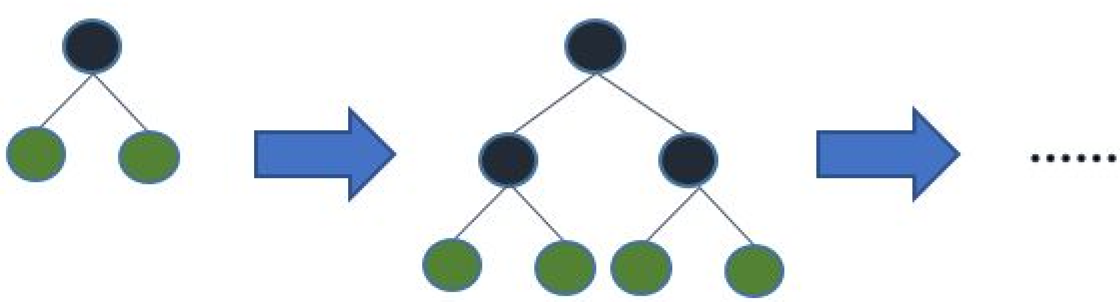
Characteristic	Job Salary Prediction and Dataset	GBM
Problem type	Regression	Supports both Regression and Classification
Data size	Small 244,000 rows, 12 columns, 412MB. After vectorization, over 400 columns.	Can process parallel well with one H2O node. (Over 10GB, can use 2 to 4 nodes; Over 100GB, can use over 10 nodes)
Missing	3 fields, ContractType missing up to 73% rows	Missing and categorical data are handled automatically without requiring any preprocessing from the user.
Sorted	No	No matter sorted or not.
Categorical feature	Yes	Missing and categorical data are handled automatically without requiring any preprocessing from the user. It internally stores the factors as integers and the column has a mapping from integers to strings.
Skewed	Highly skewed in	Has “histogram_type” parameter supports

	both the target field (Salary) and other features.	“quantilesGlobal” which the feature distribution is taken into account with a quantile-based binning (where buckets have equal population). This is slow but may help to increase accuracy.
Evaluation	Require minimum MAE	Has ModelMetricsRegression with MAE, Mean Residual Deviance, RMSLE, etc.

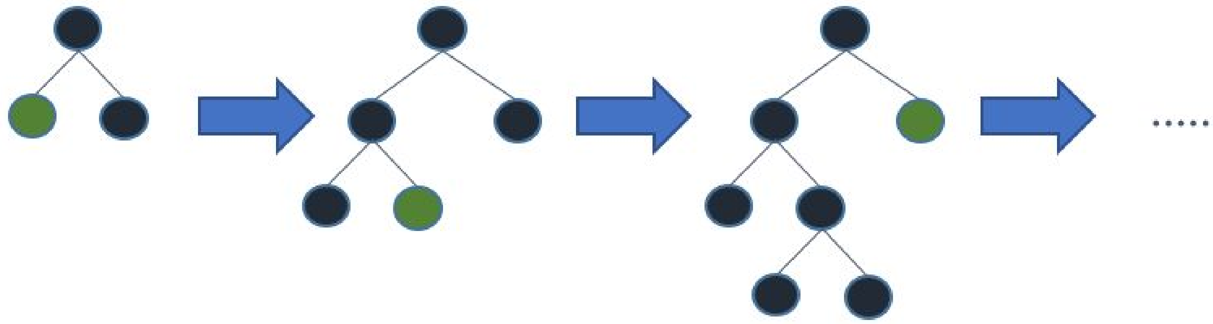
Besides GBM, there is quite a few effective implementations such as GBM, XGBoost and lightGBM. It supports well for both regression, classification, ranking and many other machine learning problems. I started with GBM but then I found XGBoost. XGBoost has become a de-facto algorithm for winning competitions at [Analytics Vidhya] and Kaggle, simply because it is extremely powerful. But given lots and lots of data, even XGBOOST takes a long time to train. LightGBM can solve this weakness.

Light GBM is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm. It is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise. So when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word ‘Light’.

Below are comparison between level-wise tree growth (XGBoost) vs Leaf-wise tree growth (LightGBM).



Level-wise tree growth



Leaf-wise tree growth

Leaf wise splits lead to increase in complexity and may lead to overfitting and it can be overcome by specifying another parameter max-depth which specifies the depth to which splitting will occur.

Advantages of Light GBM

1. Faster training speed and higher efficiency: Light GBM use histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
2. Lower memory usage: Replaces continuous values to discrete bins which result in lower memory usage.
3. Better accuracy than any other boosting algorithm: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
4. Compatibility with Large Datasets: It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.
5. Parallel learning supported.

So I will focus on lightGBM.

Benchmark

Split data into train, validation, and test set with ratio 0.7, 0.15, 0.15 to train, validate and test to have a good model can predict well for unseen data.

Use Grid search and cross-validation to find the best algorithm and its parameters which gives the smallest MAE on the test dataset.

Besides looking at the main metric (MAE), I will check the History score graph to see overfit or underfit issue to adjust parameters or even algorithm or input features.

Beside MAE, calculate the Good rate on test dataset to see how many percentages of jobs the engine can predict with acceptable differences. The higher the better.

I will plot the histogram of percent variance on test dataset to see how good/bad the model is.

Compare MAE with the competition's Leaderboard (Public)
https://www.kaggle.com/c/job-salary-prediction/leaderboard
A short board here:

#	Score	#	Score
1	3,464	60	6,066
10	4,317	70	6,250
20	4,974	80	6,413
30	5,405	90	6,569
40	5,637	100	6,717
50	5,825		

If deploy this engine to production, need to feed new job ads with salary periodically (daily or weekly) to have new data, retrain, evaluate and re-deploy.

One very good way is asking users about the accuracy of the prediction for each job on the website. With high accuracy predicted jobs from feedback, we can double them in the dataset for next train and analyze more their feature to see any special. With low accuracy predicted jobs, we need to investigate to improve.

Methodology

Data Preprocessing

Below is the summary of pre-processing tasks:

Task	Fields	Method/Tool
Categorize structured data	6 fields:	H2O import_file() with col_types parameter

	ContractType, ContractTime, Company, LocationNormalized, Category and SourceName	
Vectorize unstructured data	2 fields: Title and FullDescription	H2O word2vec model H2O tokenize() Filter out stop words

The dataset has over 244k jobs, 12 columns. Besides the 'Id' field and the target field ('SalaryNormalized') are not used as a features, there are 2 raw fields, LocationRaw and SalaryRaw, I will ignore in this capstone (because their normalized fields look good and I don't have much time to analyze them). So I have 8 remaining fields to use as input features.

There are 6 structured features: ContractType, ContractTime, Company, LocationNormalized, Category and SourceName. I will use them as categorical (enum) type.

There are 2 unstructured fields, Title and FullDescription.

FullDescription is a long text field, many jobs have over 200 words. I'll filter out common words (e.g. "there","all","we","one","the","a", etc.) then vectorize them using H2O's Word2vec to have good latent features. Transform them to vectors representing the important meaning of job title and full description well in the way easy to feed into a machine to learn.

Title is a short text field, just 1 to 10 words. They are clean job titles, not contain highlight words (e.g. "Urgent", "Hot", "High Salary", etc.) or symbols (e.g. *,! ☆, etc.). There are 135,436 unique Jobs titles. So I tried 2 solutions: vectorize like FullDescription and convert to categorical type.

Implementation

I use H2O library on Jupyter notebook with Python 3.6 kernel.

Below is the structure summary of my capstone implementation:

Context	Name/Purpose	Description
Constants	Define constants	There are over 10 constants, they are for: - Stop words to filter out - Seed to have same starting conditions in alternative configurations. - Word vector size, epochs to train. - File paths for dataset, transformed data frames, trained model, etc. - Good rate.
Functions	Small utility functions	To print time in nice format to know run time and whether the a result of a cell is new or old.

	load_data	Load data in the input file into H2OFrame, with desired column type. Input: data_file Output: H2OFrame
	tokenize	Tokenize a string column of a H2O data frame, filter out too short words, numbers and too common words. **Input**: A H2OFrame of a string column **Output**: A H2OFrame with a single column representing the tokenized, filtered Strings
	vectorize_title	Vectorize for Title column. Input: A H2OFrame contains a Title string column and others. Output: A H2OFrame contains a vectorized Title column and others. Steps: - Tokenize strings into sequences of words - Build Word2vec model - Transform sequences of words to vector - Replace original string column by the new vector column.
	preprocess	Vectorize for Title and FullDescription. Has option to vectorize FullDescription but categorize Title. Save processed frames to file. Input: A H2OFrame contains Title & FullDescription in string. Output: 2 H2OFrames, one contains vectorized Title and vectorized FullDescription, the other contains enum Title and vectorized FullDescription. Save processed frames to files.
	split	Split the input dataset into train, validation and test data set with the ratio: 0.7 : 0.15 : 0.15 (I also tried 0.6 : 0.2 : 0.2)
	grid_init_GBM, grid_init_XGBoost, grid_init_RF	Init grid search with parameters and hyper parameters for each algorithm. Input: none Output: H2O grid search.
	grid_train	Grid search to find optimum parameters

		Support 3 algorithms: H2OGradientBoostingEstimator, H2OXGBoostEstimator and H2ORandomForestEstimator. Input: train_df, valid_df and algorithm Output: H2OGridSearch
	show_score_his	Show scoring history to know the model is underfit or overfit. Input: H2O model, interateType (Trees or Epochs) Output: plot
	evaluate	Evaluate Good rate and MAE of model on test data Input: H2O model, H2OFrame Output: Good rate, MAE
	train_evaluate	Train and evaluate model, save all parameters of the best model to file. Input: H2OFrame Output: Good rate, MAE, H2O Model
MAIN	Real run	- Init H2O cluster (or connect to the running) - If USE_SAVED_FRAME: load processed H2OFrame from file, else load dataset from file then call preprocess(). - If USE_SAVED_MODEL: load trained model from file, else call train_evaluate() then save the best model to file. - Show the best model (Metrics, Cross-validation, Variable importance) - Stop H2O cluster if specified.

Refinement

Through the Kaggle competition I knew Deep Learning is the winner, and Random Forest algorithm was also used popularly among competitors. I started this project with a basic GBM then tried basic Deep Learning and Random Forest (RF). They took hours to run. I saw GBM had better result and faster. I will learn and try more with Deep Learning in another project. RF takes **random** samples of data, build learning algorithms and take simple means to find **bagging** probabilities. GBM has the selection of sample is made more **intelligently** and subsequently give more and more weight to hard to classify observations. GBM plays a crucial role in dealing with bias variance trade-off (controls both the aspects, bias and variance) so it's considered to be more effective than only controlling for high variance of RF. That's why I focus on GBM. Then I found XGBoost which is better. Finally I saw lightGBM is the best for this project.

Vectorized data type vs String data type

I tried GBM, compared vectorized and string columns (Title and FullDescription) on test dataset. Below table shows vectorization with Word2Vec improves the precision of engine significantly, 23% MAE. MAE here is high because GBM was not optimized yet. So later try I only use vectorized dataset.

	MAE	Good rate
String	11,250	66
Vector	8,652	77

(see detail in the exported notebook, predict_sal-kaggle-vec-better-novec-good-75%25.html)

I also tried to compare vectorized Title vs categorical Title and see vectorized is better too. Unfortunately I didn't record the result. Then later I only used vectorization for both Title and FullDescription.

Summary: Vectorization with Word2Vec improves MAE (23% in my test).

Grid search

Grid search is a very good tool of H2O help us easily find out the optimal parameters.
However, when the running time increases, over 4 hours to have a model, it becomes difficult to wait for the result and sometime H2O can die after a long and heavy run (over 8 hours).
So I just have the result of grid search of rather simple models (low accuracy).
Below is the Grid search result with DRF and GBM, showing more trees and higher depth improve Deviance (also improve MAE).
DRF

	max_depth	**ntrees**	**model**	**residual_deviance**
0	30	100	model_3	1.1408188996691054E8
1	30	50	model_1	1.1590589666682549E8
2	15	100	model_2	1.225563553278003E8
3	15	50	model_0	.237955779342987E8

GBM (learn_rate 0.1, col_sample_rate 0.6)

	max_depth	ntrees	model	residual_deviance
0	9	350	model_3	9.334347088155913E7
1	9	300	model_1	9.39566319134783E7
2	8	350	model_2	9.398591149148375E7
3	8	300	model_0	9.481583404133098E7

I also ran a grid search with DRF showing the more trees the worse precision (please check "predict_sal-20171123-RF-good75-mae9087-runslow.html" file)

LightGBM turning

Parameter	Meaning	Apply
Learning rate	The bigger learning rate the faster convergence. Default 0.3.	Try default first, decrease to improve the accuracy. 0.02 is the best suit for my project.
ntrees	Specify the number of trees to build. This value defaults to 50.	I tried the default value, then increase to 100,200, 500, 800, 1000, 1200. Finally 1000 is the best.
col_sample_rate	Specify the column sampling rate (y-axis) for each split in each level. Default 1. Can use to speed up as well as prevent overfit.	Try default (1) then reduce to 0.4 for fast and 0.6 is the final.
max_leaves	Specify the maximum number of leaves to include each tree. This is the main parameter to control the complexity of the tree model. should let it be smaller than `2^(max_depth)` Default 0.	I tried GBM, XGBoost before lightGBM and found max depth 10 is good. When switch to LightGBM, I tried max_leaves is less than 1024 (2^10) and 800 is the best (with max_depths is 0).
max_bins	specify the maximum number of bins for binning continuous features. This	Use small max_bins for fast and prevent overfit. Final is 128

	value defaults to 256	
nfolds	Specify the number of folds for cross-validation. default 0, 5-10 is good but 10 will take more time.	I tried 5 and 4 then 5 is the best.
stopping_rounds, stopping_metric	Stops training when the option selected for stopping_metric doesn't improve for the specified number of training rounds, based on a simple moving average. This value defaults to 0 (disabled)	I use 3 which should be smaller than nfolds (5). I use MAE as stopping_metric, same with the metric of this problem.

Some other takeaways

Item	Takeaways	More explanation
H2O estimated remaining time	For long H2O jobs, need to wait for 3-5 minutes after starting to see the stable remaining time estimation.	After starting 1 minute, many times H2O shows estimated remaining time over 20 hours then after 5 minutes it shows 4 hours.
H2O connection	Not run 2 notebook connect to one H2O cluster with only one node.	H2O will be very slow then die.

Results

Model Evaluation and Validation

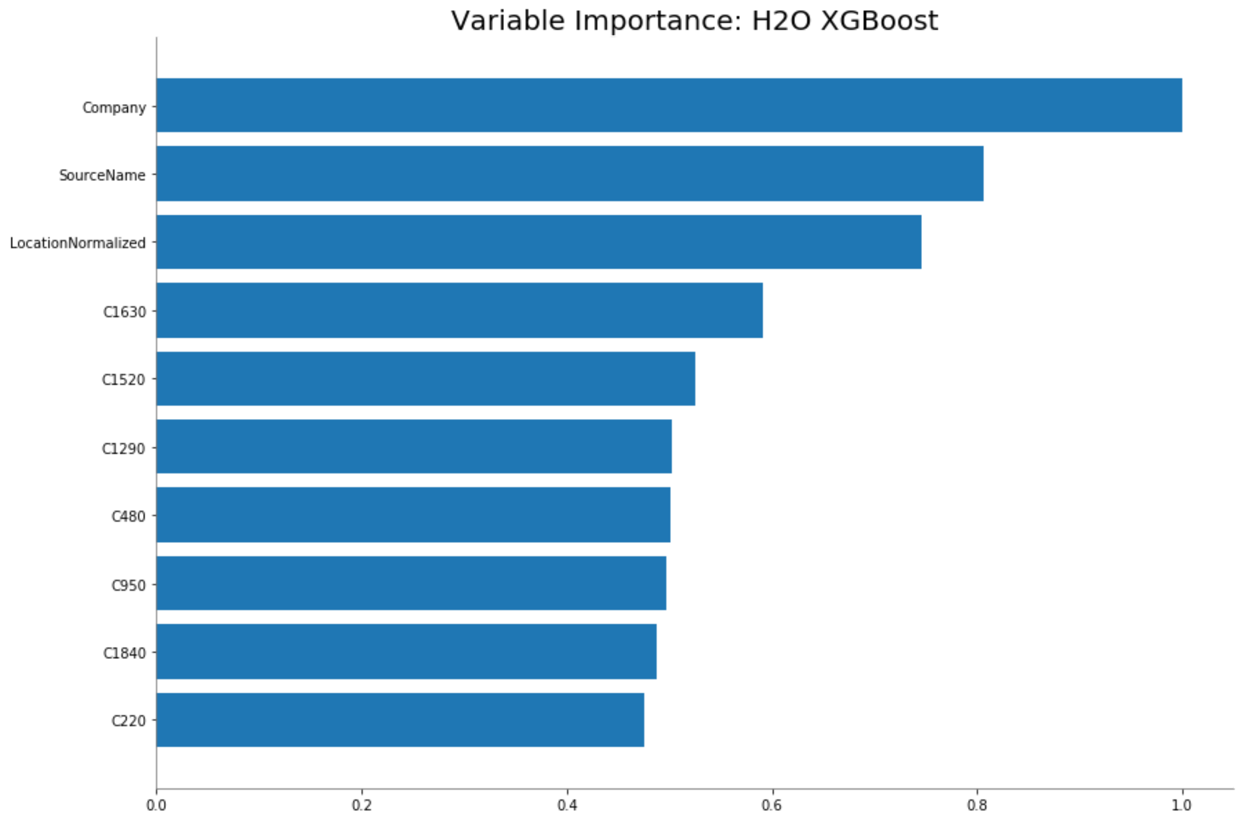
Final result summary (the best model):

What	Information/Result
Problem	Job Salary Prediction. A regression supervised learning problem.
Dataset	244,768 jobs, 12 columns with 412MB compressed size. The dataset is highly skewed to the left, 75% of jobs have the lower salary than 25% of max salary. Have a rather big text field.
Metric	MAE: **5761** Good rate: 83
Algorithm	Use **LightGBM** on H2O AI platform via XGBoost machine.
Software	H2O cluster version: 3.14.0.3 Python version: 3.6.2 final Jupyter version: 4.3.0
Hardware	8 Cores 12G RAM Macbook pro
Runtime	Around 6.5 hours. This is not included the vectorizing 2 text fields (around 0.5 hours)
Parameters	Learning rate: 0.02 Max leaves: 800 (max_depth: 0) Col sample rate: 0.6 Nfolds: 5 Stopping round: 3 Stopping metrics: MAE

More analysis about choosing parameters is mentioned in the Refinement part.

Please check this exported file for reference: predict_sal-good83-mae5761-trees1000-xgboost.html

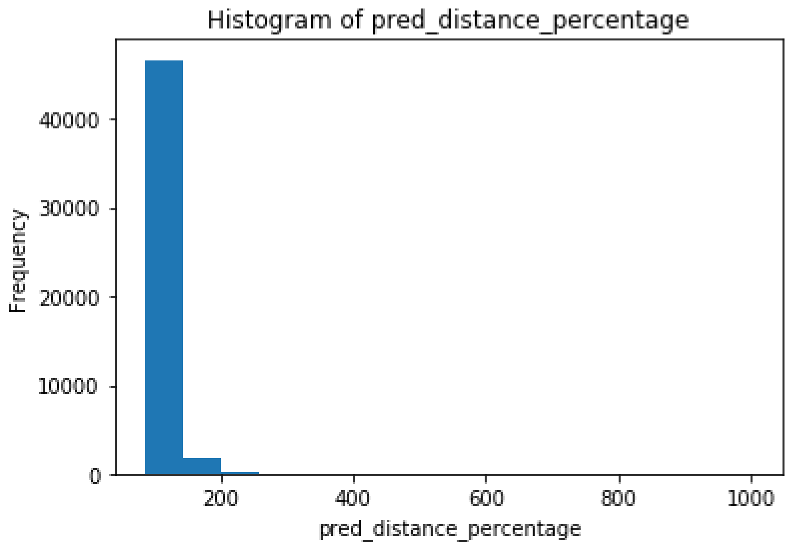
The model can detect the top 3 most important attributes are Company, SourceName and LocationNormalized. This is reasonable because a CEO of a small local company may still have much lower salary than a Manager of a global company; some job board sites have only high salary jobs while others have only medium or low salary jobs; expensive cities have much higher salary jobs than small cities.



Justification

The best MAE score I got on the validation data is **5761.7** which is between the position 41th and 42nd in the Public Leaderboard, 31st and 32nd of the Private Leaderboard (https://www.kaggle.com/c/job-salary-prediction/leaderboard). **Generally, in top 50**. And the Good rate is **83/100** which means the engine can predict the salary (accept +- 30%) of job based on other information with the accuracy over 80%. I'm satisfied with this result for now.

Good pred %: 83.0 (40440 / 48831)



Conclusion

Free-Form Visualization

I will show some jobs in the test dataset and their prediction result.

combined-prediction-327a4313-e0bb-4d7b-90e5-3e08c413146b

▾ DATA

← Previous 20 Columns → Next 20 Columns

Row	C1	LocationNormalized	ContractType	ContractTime	Company	Category	SalaryNormalized	SourceName	C10	
1	36654.7969	Dorking	?	permanent	Gregory Martin International	Engineering Jobs	25000.0	cv-library.co.uk	-0.0373	0.2
2	25336.8984	Manchester	?	permanent	Code Blue Recruitment	HR & Recruitment Jobs	22000.0	cv-library.co.uk	-0.0391	0.2
3	16986.1523	Derby	?	?	Chef Results	Hospitality & Catering Jobs	16000.0	caterer.com	0.0892	0.0
4	33922.1328	Wolverhampton	?	permanent	Rullion Engineering Ltd	Engineering Jobs	33500.0	cv-library.co.uk	-0.0161	0.1
5	36630.0117	Newcastle Upon Tyne	?	permanent	Asset Appointments	IT Jobs	21000.0	cv-library.co.uk	0.0071	0.1
6	32231.8770	Cambridge	?	permanent	Indigo 21 Ltd	IT Jobs	30000.0	cv-library.co.uk	-0.2278	0.1
7	28135.5781	UK	?	permanent	Kirklees Active Leisure	Travel Jobs	25776.0	leisurejobs.com	0.1115	0.4
8	26426.6855	UK	?	contract	Kirklees Active Leisure	Travel Jobs	20880.0	leisurejobs.com	0.1562	0.4
9	32944.8711	UK	?	permanent	Indigo 21 Ltd	IT Jobs	35000.0	cv-library.co.uk	-0.3567	-0.0

The first column, ‘C1’, is the predicted salary and the actual salary is ‘SalaryNormalized’.

We can see a job of ‘Chef result’ company in Derby is predicted 16,980 while the true salary is 16,000. Another job of ‘Indigo 21 Ltd’ company in UK has 32,944/35,000. Only 6% deviance, good, right? These good predictions provide meaningful information to user.
The result is event better for some below cases. A job in Healthcare & Nusing in Blyth has 24,099/24,000, the deviance is under 0.5%!

25	27952.5781	Nottingham	full_time	?	?	Healthcare & Nursing Jobs	26000.0	careworx.co.uk	0.0659	0.16
26	24099.4668	Blyth	full_time	?	?	Healthcare & Nursing Jobs	24000.0	careworx.co.uk	-0.4315	0.12
27	30537.9785	Leicester	full_time	?	?	Healthcare & Nursing Jobs	28000.0	careworx.co.uk	-0.1116	0.04
28	33173.2891	Scotland	?	?	?	Healthcare & Nursing Jobs	67200.0	careworx.co.uk	-0.1227	-0.12
29	33847.9922	Scotland	?	?	?	Healthcare & Nursing Jobs	33600.0	careworx.co.uk	-0.0597	0.14
30	38215.2344	Gainsborough	full_time	?	?	Healthcare & Nursing Jobs	42500.0	careworx.co.uk	-0.0513	0.03
31	39884.5586	Hadleigh	full_time	?	?	Healthcare & Nursing Jobs	42500.0	careworx.co.uk	-0.0170	0.01
32	20345.9805	Buckinghamshire	part_time	?	?	Healthcare & Nursing Jobs	20160.0	careworx.co.uk	0.1082	0.11
33	47648.0	London	full_time	?	?	Healthcare & Nursing Jobs	48000.0	careworx.co.uk	0.1235	0.05

Some jobs has deviance over 40%, these predictions are hard to accept by users. But overall, the engine can predict accurately for 83% jobs with max 30% deviance as showed in Justification part and I think it is acceptable by users for an alpha release of the Job Salary Prediction feature of a website.

Reflection

Project task summary as below:

Step	Task
1	Research for the problem related to my work with a good dataset. Collect the domain knowledge.
2	Analyse the dataset and research the solutions.
3	Preprocessing data.
4	Split the dataset to have train, valid and test sets to prevent overfit.
5	Research then try different algorithms/models and compare on the metrics, run time.
6	Make the proposal report.
7	Research more and try more to tune the performance for the best algorithm with

	different parameters.
8	Evaluate and make the report

Every steps is interesting and help me learn a lot. The most difficult step is 7 which takes a lot of time to see a result. There is the grid search solution to easily find out the best parameter set, but the hardware limitation takes long time to see result so I just applied at the beginning which the model converges fast but low accuracy. To increase the accuracy more I need to wait longer to see the result and to prevent H2O die I need to run single parameter set and compare their performance manually. If I have free GPU to train faster it'd be great.

Improvement

The engine can predict job salary rather good, 83% accuracy with the acceptance of max 30% smaller or bigger the true value. And with the MAE 5761, my engine is in the top 50 best engines in the Kaggle' Job Salary Prediction Competition. Certainly I want to increase the accuracy and also reduce the training time. Some ideas:

- With more powerful hardware resources like GPU or higher number of cores or multiple H2O nodes I can increase the accuracy.
    - Increase number of trees, like to 1200
    - Reduce learning, like to 0.01
- The salary range of the dataset is too big, from 5,000 to 200,000 and the distribution is very skewed (75%(q3) jobs have under 43,000, which is smaller than 25% of max SalaryNormalized). When have more time I may split the dataset into 2 groups, under 43,000 and from 43,000 to analyse and train separately then use both models to feed into another model to have a final better result.
- Try CNN, the proven great algorithm.

References

[1] Kaggle Inc., "Job Salary Prediction", 2013. http://www.kaggle.com/c/job-salary-prediction
[2] Guolin Ke and others, "LightGBM: A Highly Efficient Gradient Boosting Decision Tree", 2017.
https://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf
[3] Pranjal Khandelwal, "Which algorithm takes the crown: Light GBM vs XGBOOST?", 2017.
https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/
[4] Microsoft Corporation, "GPU Tuning Guide and Performance Comparison", 2017.
http://lightgbm.readthedocs.io/en/latest/GPU-Performance.html
[5] S. Jackman and G. Reid, "Predicting Job Salaries from Text Descriptions", 2013.
https://open.library.ubc.ca/cIRcle/collections/42591/items/1.0075767.

[6] NSS, “An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec”, 2017. https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.
[7] H2O.ai, "Word2vec", 2017.
http://docs.h2o.ai/h2o/latest-stable/h2o-docs/data-science/word2vec.html
[8] Navdeep Gill, “A Word2vec demo in Python using a Craigslist job titles dataset”, 2017.
https://github.com/h2oai/h2o-3/blob/master/h2o-py/demos/word2vec_craigslistjobtitles.ipynb.
[9] Aarshay Jain, "Complete Guide to Parameter Tuning in Gradient Boosting (GBM) in Python", 2016.
https://www.analyticsvidhya.com/blog/2016/02/complete-guide-parameter-tuning-gradient-boosting-gbm-python/
[7] Microsoft Corporation, "Parameters Tuning", 2017.
http://xgboost.readthedocs.io/en/latest/model.html
http://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html