Machine Learning Engineer Nanodegree
Capstone Proposal

Ninh Sai
November 26th, 2017

Domain Background

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.

As my knowledge, Job Level will affect salary very much, the higher level the higher salary. CEO must have much higher salary than Entry Level.

Year of Experience also affects but less than Job Level. 10 years of experience but still only a Manager has a lower salary than 5 years but CEO.

Company also affects to salary a lot. A CEO of a small local company may still have much lower salary than a Manager of a global company.

Company size, normally the bigger size the higher salary, but this will be more exact if compared companies are in the same industry.

Industry, an Experienced IT position may have a higher salary than a Legal Manager.

Location, the salary of jobs in a big expensive city is higher than this in a small-poor city.

Skill also helps, Java skill is offered much higher salary than DevOps.

Job Title and Job Description can help in case some or all above fields are missed or not well used.

Job Title can contain Job Level (CEO, Manager, Experienced, Entry, Senior, Junior, etc.), Skill (Java, PHP, Oracle, etc.). In some job board sites, many employers put years of experience, company name, location, benefit and even salary in Job Title to attract job seekers. Sometimes, higher urgency level of a job can increase salary, and employers often put in Job Title too.

Job description can specify Highest Level of Education (Ph.D., Master, Bachelor, etc.), Benefit (MacBook Pro, expensive health care package), Growth chances.

Additional characters in Job Title and Job Description also change salary range.

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

Build an engine to predict job salary, 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.

I use Kaggle's Job Salary Prediction dataset to train and test.

I need to build an engine can do preprocessing data set, train then predict on unseen data.

The well-known evaluation metric for this problem is Mean Absolute Error (MAE).

Beside MAE, I also calculate the ratio of good predictions/total predictions, which good prediction is predicted salary is not different from real salary more than 30%.

I'll try different algorithms with different parameters to have a minimum MAE and a high good ratio.

This competition has a Leaderboard with the score is MAE. I can compare my result with their scores to know how good level my solution is.

There are some algorithms can solve this regression supervised learning problem. The winner of the competition, Vlad Mnih, use his own deep neural networks as this Q&A. I love H2O, the world's leading open source AI platform. I tried some simple deep learning of H2O but got a bad result (need more time for me). I found Gradient Boosting Machine (GBM), Word2Vec estimators good for this regression problem so I'll try more with them.

Datasets and Inputs

Download the dataset

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

Data description

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.

Data analysis

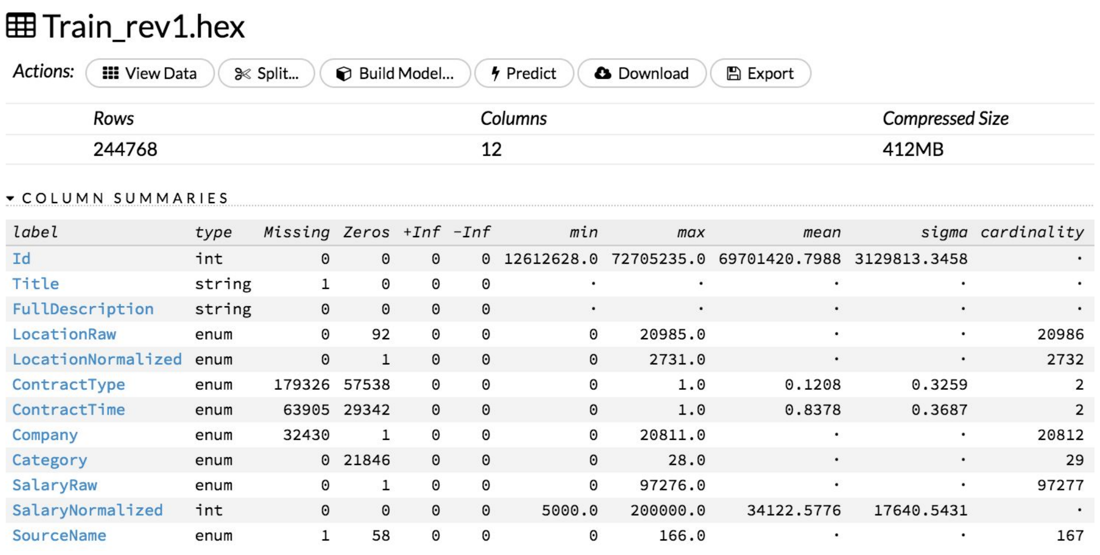
H2O is excellent at the user interface. I use its web-based interactive environment, H2O Flow, to analyze the dataset.

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

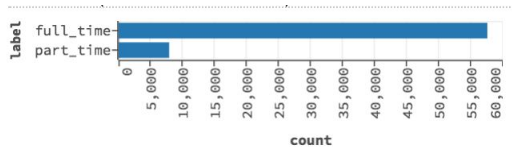
Actions: View Data | Split... | Build Model... | Predict | Download | Export

Rows	Columns	Compressed Size
244768	12	412MB

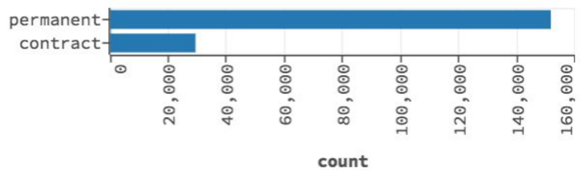
COLUMN SUMMARIES

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

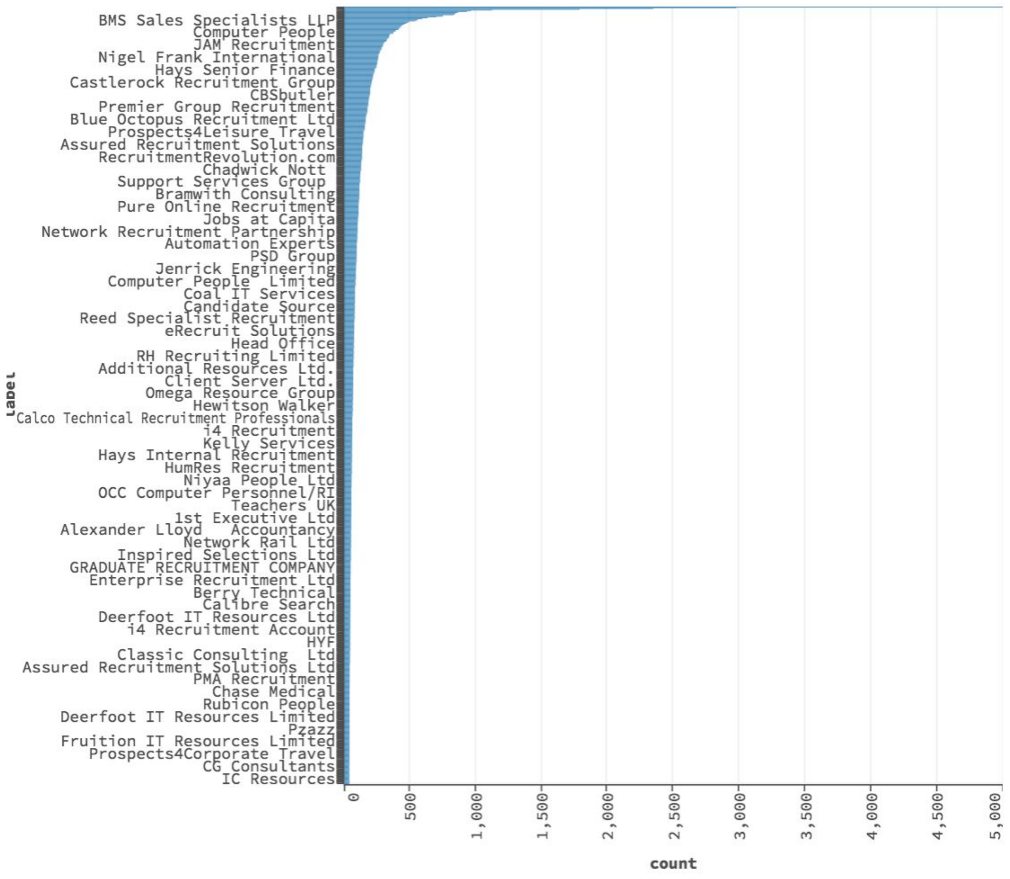
- Title and FullDescription are strings and FullDescription has long strings causing big size for the dataset. Only 1 jobs missed Title. These fields are important to use.
- LocationRaw has 20,986 unique values but Adzuna normalized to only 2732 LocationNormalized unique values. I will use LocationNormalized 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.

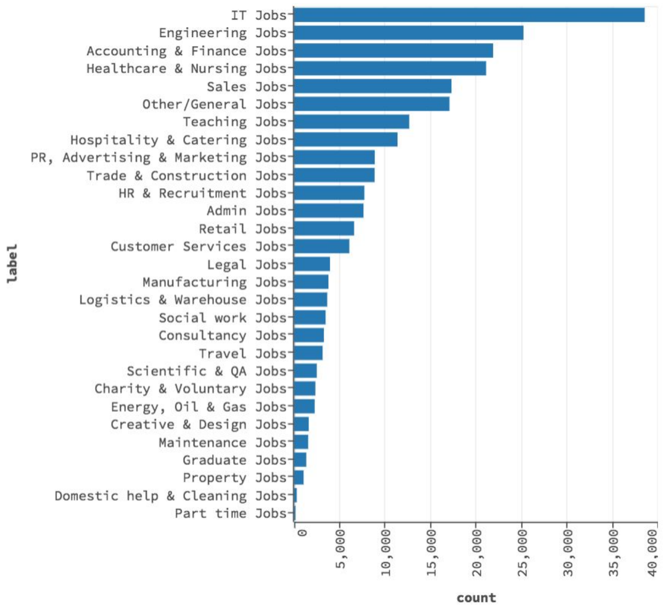


- Company: There are 20,812 companies. 13% jobs have no company information.

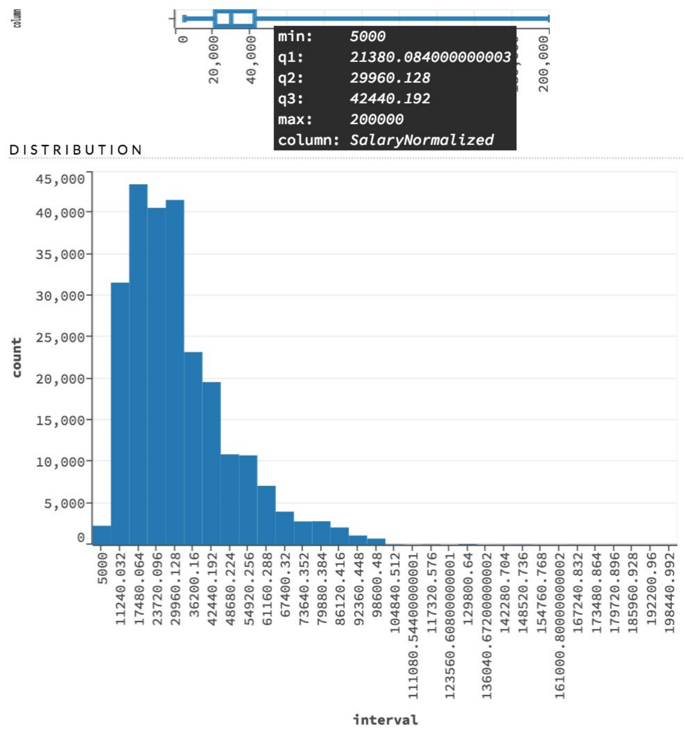


- Category: 29 categories. No missing.

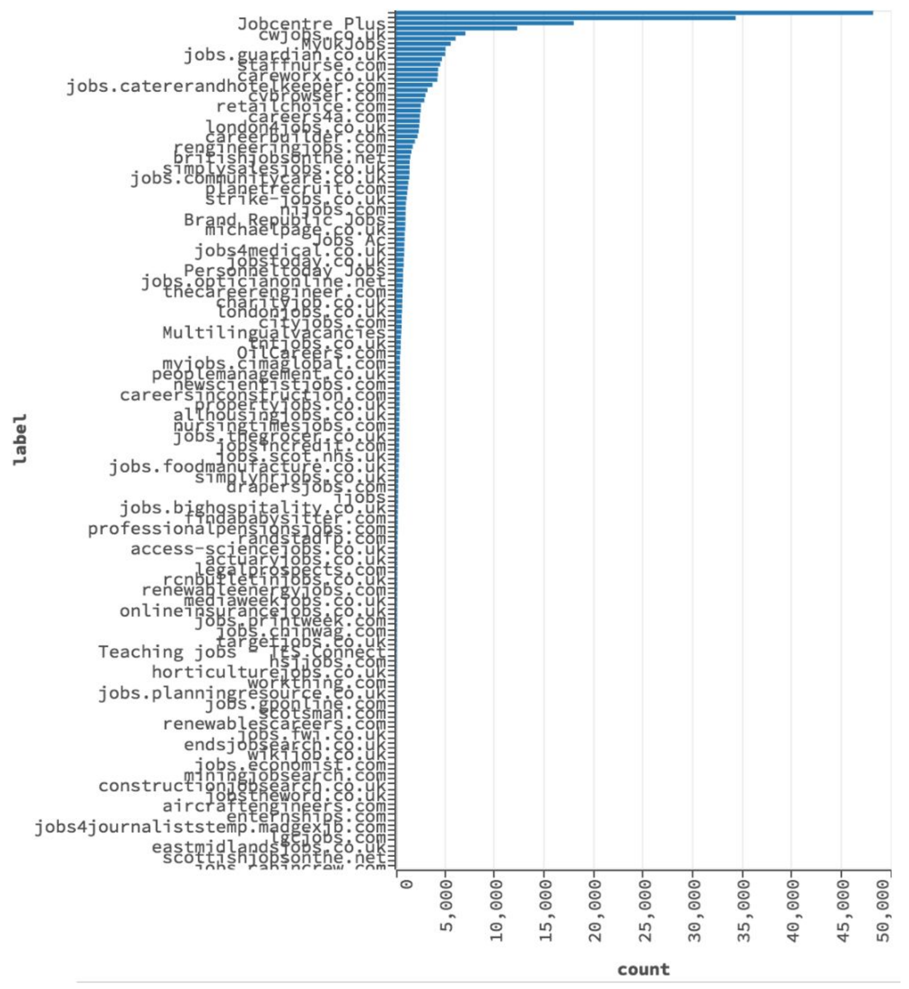
Top 3 categories are: “IT”, “Engineering” and “Accounting & Finance”. Their jobs occupy 35% total jobs (85,503/244,768).



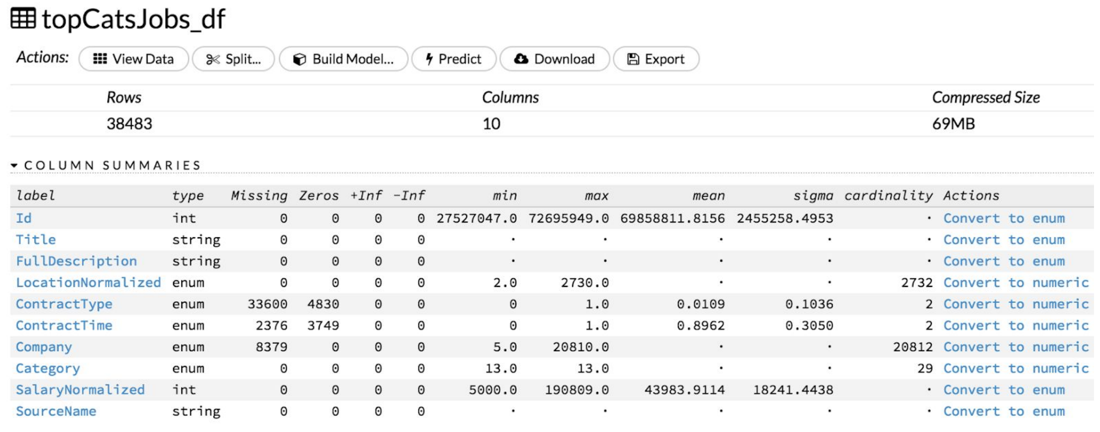
- SalaryRaw is text so ignore this time.
- SalaryNormalized: min 5,000, max 200,000 and mean about 34,000.
  The below distribution chart shows SalaryNormalized is high right skew. 75%(q3)
jobs have under 43,000 (which is smaller than 25% of max SalaryNormalized).



- 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
```

Solution statement

Solution

	Solution/Approach
Dataset	Kaggle's Job Salary Prediction Train dataset
Evaluation	Mean Absolute Error (MAE). Minimize MAE. Good rate. Maximize Good rate. Compare MAE with the competition's Leaderboard
Algorithm	Try to use some powerful algorithms for regression then select the best. The top ones I know are Deep Learning, Random Forest, Gradient Boosting Machine, XGBoost. Use Word2Vec to have good latent features from the big text field (FullDescription), consider applying for Title field too.
Benchmark	Split data into train, validation, and test set with ratio 0.7,0.15,0.15. Grid search and cross-validation. Compare MAE with the competition's Leaderboard. Use Scoring history graph to know the model is good, overfitting underfitting then turning.
Library/Platform	H2O, Jupyter notebook, Python 3.6

Analysis leads to solution

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, 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).

H2O has 6 over 7 supervised algorithms support regression problem. They are Deep Learning (Neural Networks), Distributed Random Forest (DRF), Generalized Linear Model (GLM), Gradient Boosting Machine (GBM), Stacked Ensembles and XGBoost.

I'd like to try about 3 different algorithms and find the best fit model based on MAE and Good rate in the reasonable time and hardware resources of mine.

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

I'll use H2O's Craigslist job titles example as my start, for both preprocessing and training with GBM. Then I will try others, compare then choose one best fit.

H2O is really fast and powerful for me to use. I don't list its great features, just add links so anyone can explore fully.

More important is it's used by top companies in the world like Cisco, eBay, Paypal and Comcast and more. I hope I have a good result for Job Salary Prediction with H2O.

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 both the target field (Salary) and other features.	Has "histogram_type" parameter supports "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	Has ModelMetricsRegression with MAE, Mean

	MAE	Residual Deviance, RMSLE, etc.

Benchmark Model

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.

Evaluation 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%].

Project Design

This project will follow the typical flow to build a machine learning engine for the regression problem.

Step	Task
1	Research demand and domain background. Salary is a sensitive information any job seeker wants to know while many employers still don’t public with job ads. Predict salary based on public information about jobs is an interesting and meaningful problem. Analyse public fields of job ads and their affection to salary and feature correlation. This is a regression problem.

2	Collect data. Use Kaggle’s Job Salary Prediction Train dataset to ensure the input is good and legal. One more benefit is I can compare the accuracy of my solution with others.
3	Analyse dataset. Use H2O Flow and Jupyter notebook to have a good summary of the dataset as well as a deep understanding of each feature and feature correlation.
4	Preprocess dataset. Remove bad data, outliers. Vectorize text fields. Categorize fields with a set of values.
5	Split data into train, validation, and test set with a good ratio. Gradient Boosting Machine is a good algorithm for regression problem. Beside this Create engine, train, predict and evaluate. Looking at Scoring History to see overfitting/underfitting to decide the turning direction.
6	Continuous improve. 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. Asking users about the accuracy of the prediction for each job on the website and use feedback to improve the accuracy of the engine.

References

[1] Kaggle Inc., “Job Salary Prediction”, 2013.
[Online]. Available: http://www.kaggle.com/c/job-salary-prediction

[2] S. Jackman and G. Reid, “Predicting Job Salaries from Text Descriptions”, 16-Apr-2013.
[Online]. Available: https://open.library.ubc.ca/cIRcle/collections/42591/items/1.0075767.

[3] NSS, “An Intuitive Understanding of Word Embeddings: From Count Vectors to Word2Vec”, 4-Jun- 2017. [Online]. Available:
https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/.

[4] Navdeep Gill, “A Word2vec demo in Python using a Craigslist job titles dataset”, 7-Apr-2017.
[Online]. Available:
https://github.com/h2oai/h2o-3/blob/master/h2o-py/demos/word2vec_craigslistjobtitles.ipynb.