report_notebook_final

December 15, 2020

1 Predicting Job Salaries from Indeed.ca

1.0.1 STA 2453 Project 1

1.0.2 Tuesday, December 15, 2020

1.0.3 By: Brendan, Kexin, Cheng, Shikai

This project investigates the task of predicting jobs salaries from job advertisements collected from indeed.ca. Our two goals are 1: Create a model capable of predicting job salary and 2: Investigate which features are important to predicting job salary.

This notebook assumes that the web scraping has been performed by our web scraping scripts. We began with exploratory analysis of our scraped data, looking at distributions of the variables and then proceeded to use NLP techniques to create word features for the text data. Next, we created bag-of-words models using both our text and numerical features. To achieve our two goals we focused on interpretable models or models where we can estimate feature importance. To finish we analyzed the results. Our top logistic regression model achieved an F1 score of 0.654519. Looking at the feature importances we see that job categories and locations are the most important. After that education and key technology skills are the most important.

1.1 Introduction

If you are looking for a job it is very common to apply on a job board. Unfortunately, many job advertisers do not have the job salary information on the job ad. You want to know what you will make before you spend a large quantity of time going through through the application process and interviews. You also may want to know which factors lead to higher or lower salaries.

We have decided to investgigate two questions. Can we predict job salary based on a job advertisement? And, are there any potential features that can determine higher or lower paying salaries?

If we knew the answers to these questions we could make our job search more efficient and tailor our resume to maximize our potential salary.

We began by scraping our data from indeed.ca. Below is a short exploratory analysis of that data.

2 Exploratory Data Analysis

Before building the model to explore the relation between salaries and the data feature, we did the Exploratory Data Analysis fisrly to have a intuitve sense about these features and whether they are related to the salaries.

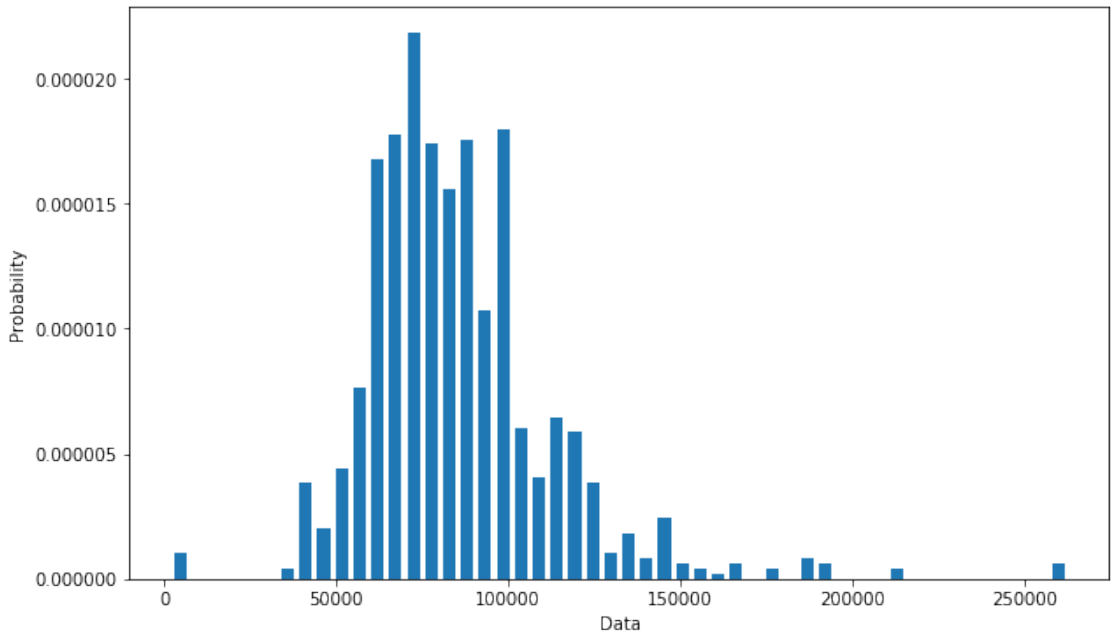
After crawling the data, we have 950 samples. Firstly, we noted that the salaries are in different forms, some are based on hour/year, and some are in a range such as 70-80 an hour, etc. Therefore, we converted them into salary based on year.

[10]:
```
0      $36.45 - $43.04 an hour
1                 $43 an hour
2                 $28 an hour
3                 $46 an hour
4               $34.50 an hour
                ...
945          $70 - $80 an hour
946            $120,000 a year
947             $40.35 an hour
948             $40.10 an hour
949                $35 an hour
Name: salary, Length: 950, dtype: object
```

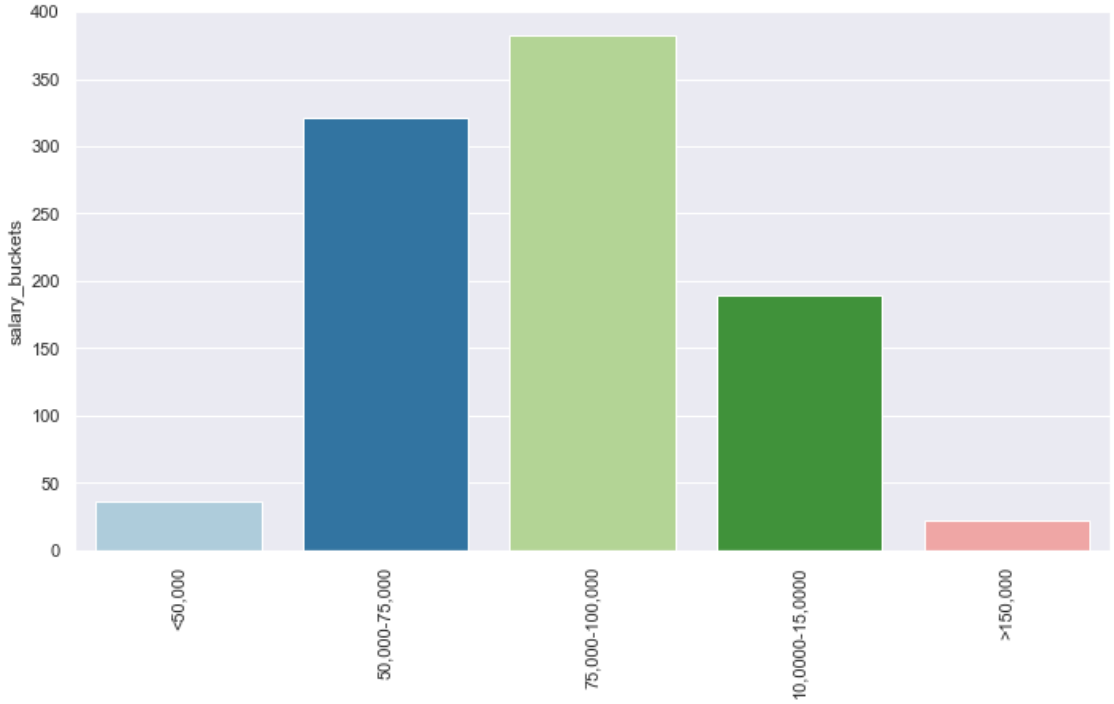
[11]:
```
0      100157.4
1      108360.0
2       70560.0
3      115920.0
4       86940.0
          ...
945    189000.0
946    120000.0
947    101682.0
948    101052.0
949     88200.0
Name: salary, Length: 950, dtype: float64
```

2.1 Distribution of the salary

We observed that most salaries are in between 50K to 100K, which agree with our intuition.



Next, We split the salaries into five ranges: <50K, 50-75K, 75-100K, 100-150K, >150K.



2.2 Preliminary analysis

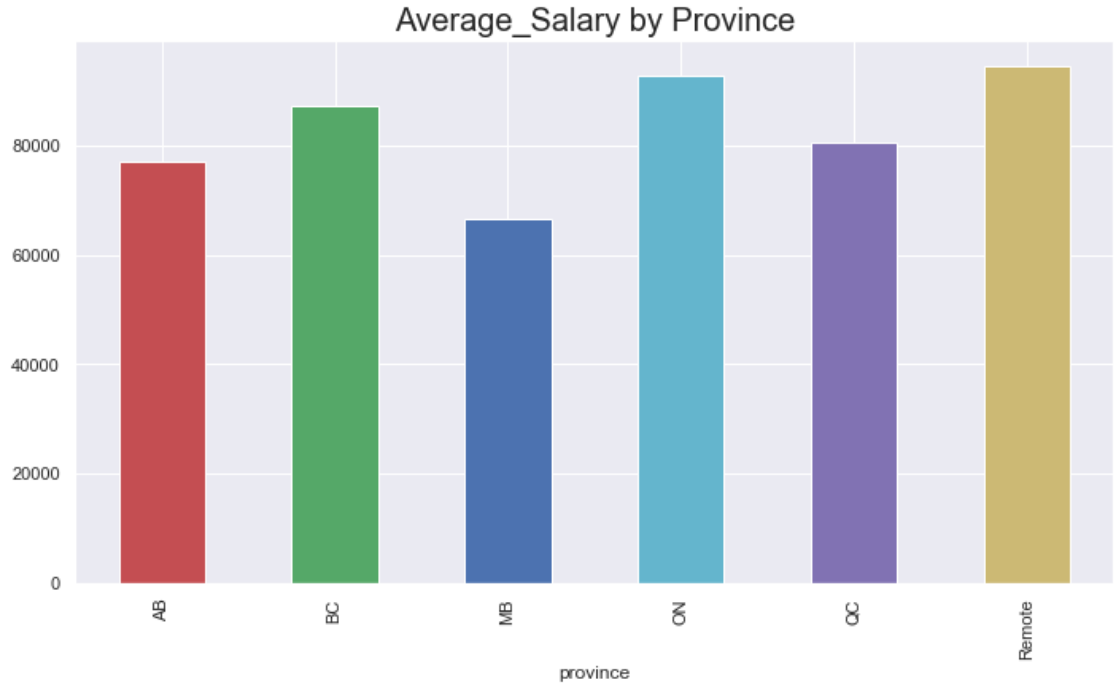
Before building models, we would like to have a preliminary analysis on the feautres.

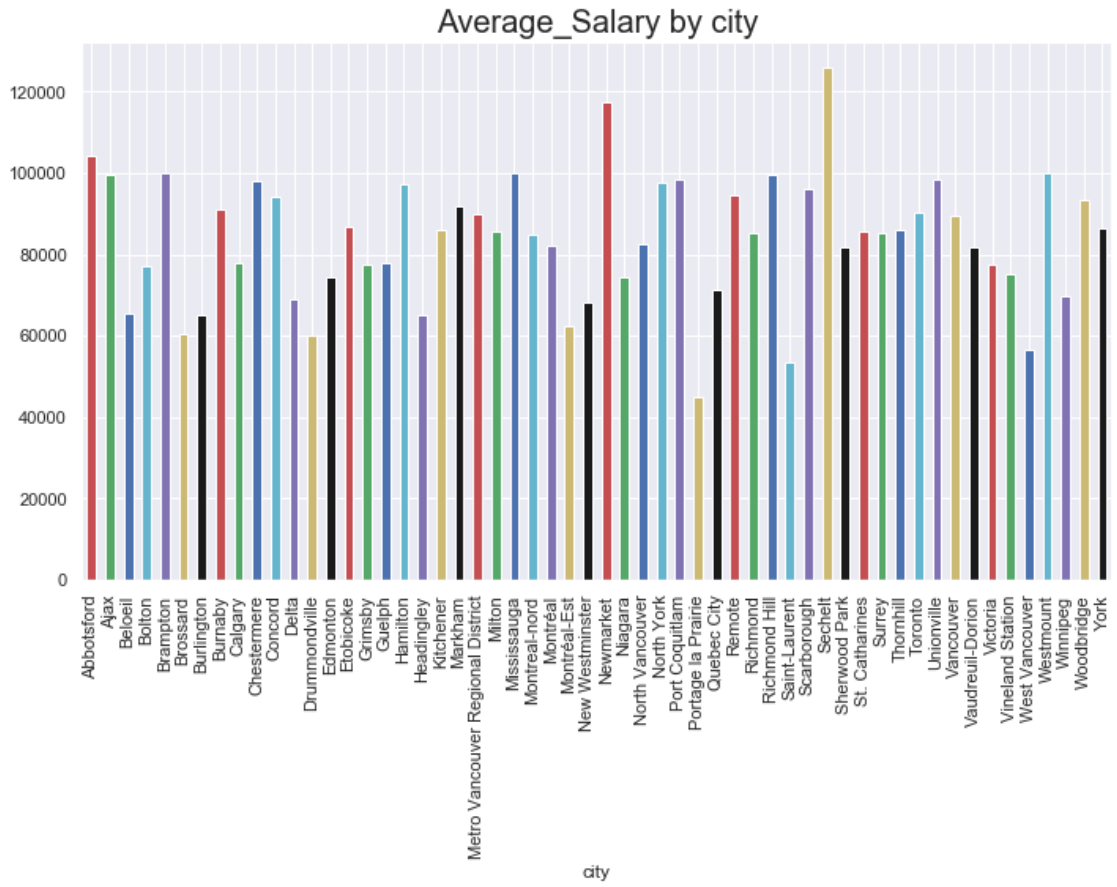
We could get a sense about how these features affect the salary by doing basic visualization.

2.2.1 Location feature

For each province and city we calculated average salaries. We observed that the average salary is different in each province. ON and BC provinces have higher salaries than other ones. In addition, average salary of remote jobs is the highest.

However, the difference of salaries among provinces is not very obvious. Instead, we looked at the average salary on each city. The difference is now obvious with some cities having average salary higher than 100K and some lower than 60K.

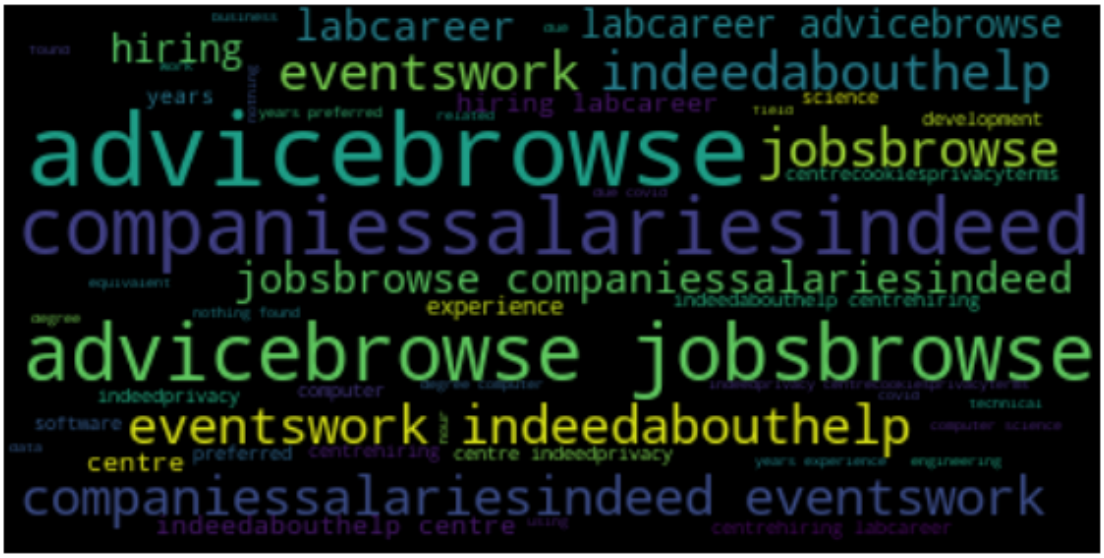




2.2.2 Job requirements feature

In order to extract features from requirement text, we needed nature language processing (NLP) method to handle with text data. Term frequency-inverse document frequency (TF-IDF) is one of the popular NLP methods, which can reflect how important a word is to a document in a corpus. Compared to bag of words, another popular NLP method, TF-IDF reduces the importance of some frequently occurring terms like we, need or have which may not be included in stop words. If a term appears in every observation, it has no contribution to the model. Thus, we decided to use TF-IDF to extract features. In python, sklearn library has a function called TFidfVectorizer. It can calculate TF-IDF of terms in each document. Because some bigrams like machine learning or deep learning may be important to predict the salary, we extracted both unigram and bigram at the same time, and we only selected top 1,000 features. If we included all features, the dimension would be too large due to small data amount.

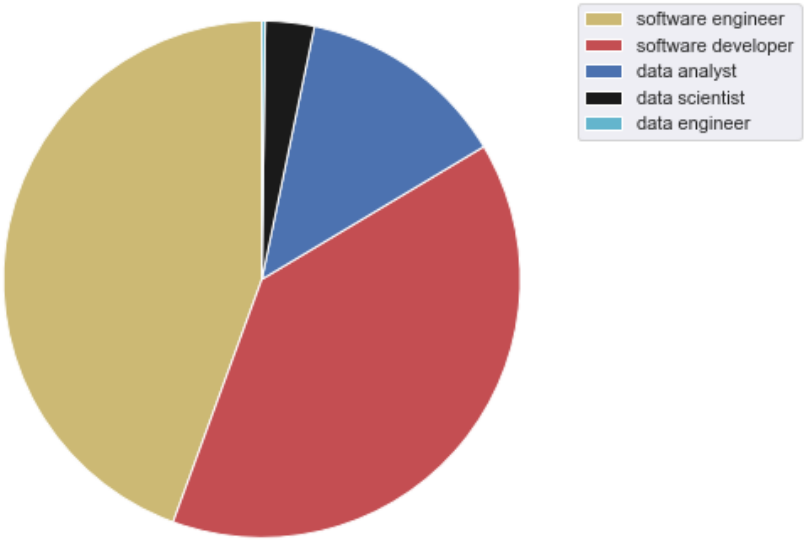
The following two images show top 50 words in selected features.



They are pretty similar no matter if the condition salary > 10K is added or not. The words like advicebrowse, companiessalariesindeed, eventswork have large overall TF-IDF values in corpus but they don't make any sense for job hunters. In this case, we need to build the models to explore the relationship between features and salary.

2.2.3 Category feature

In the data crawling step, we crawled salaries for 5 different job categories. From the figure it is clear that most jobs are still related to software and development.



Since there are not many examples of the job such as data scientist and data engineer, it is not very useful to compare the mean salary of different category, but from the scatter plots, we can still realize that most jobs from different categories are in the similar range.

2.3 Model Selection

In the project, we tried logistic regression, K nearest neighbor, and random forest, which are three popular models to solve classification problems based on the 2019 Kaggle ML and Data Science Survey. Logistic regression, as a basic classification model, is commonly used in our work so we decided to use it as one of the models to predict salary according to extracted features. In the survey, the secondary popular algorithm is tree ensemble algorithm. In addition, random forest uses bootstrap method to build trees, which can reduce the impact of imbalanced data.

Besides, it can compute the importance of features so we can know which skills affect the salary. The reason that we tried KNN is its power to solve multiclassification problems and easy to understand. In order to make sure the final model is generalized, we split the dataset into training set and test set as 80:20 ratio. For the evaluation metric, we chose weighted F1 score because the target has 5 imbalanced classes.

Compared to regular macro F1 score, weighted F1 score takes the weighted average to the value. In this case, the impact between imbalanced classes is eliminated. Additionally, we used grid search and cross validation to tune hyperparameters. First, we used job category, location and 1000 extracted features to build the models.

2.3.1 Models including job_category and location

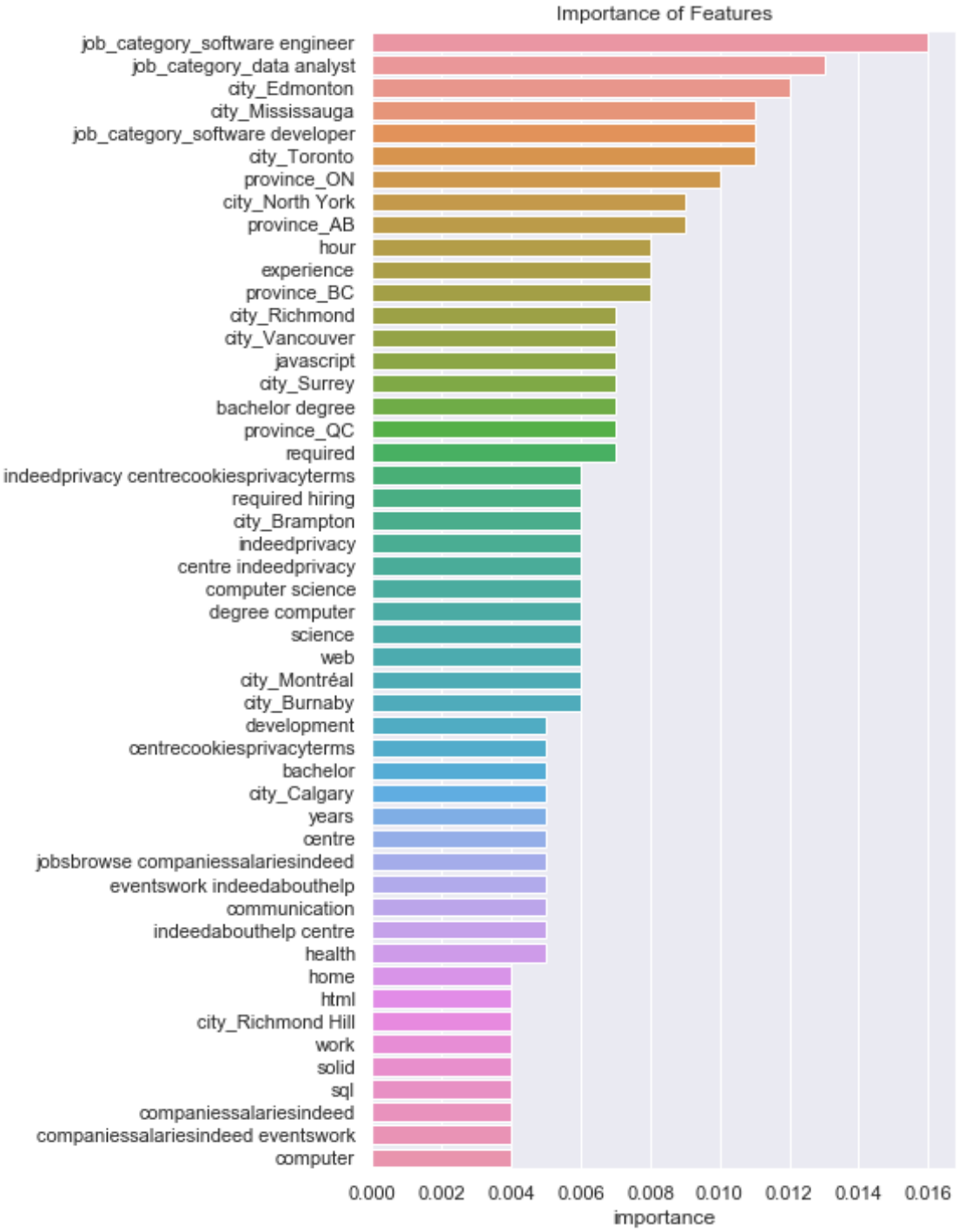
```
The optimal logistic regression model:
 LogisticRegression(C=100, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='auto', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)
==========================================================================
The mean accuracy score is 0.6718115684649897.

The optimal SVM model:
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=None, n_neighbors=10, p=2,
                      weights='distance')
==========================================================================
The mean accuracy score is 0.4350239797939933.

The optimal random forest model:
 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=50,
                        n_jobs=None, oob_score=False, random_state=44, verbose=0,
                        warm_start=False)
==========================================================================
The mean accuracy score is 0.6465241969245556.
```

[29]:

	F1 score
Logistic Regression	0.671812
KNN	0.435024
Random Forest	0.646524

```
The weighted macro f1 score of the optimal logistic regression model is
0.6791827294591143 .
```

By comparing their F1 score, we found that logistic regression model gave the highest score among these three models. On the training set, the score of logistic regression was 67.18%. On the test set, the score of the logistic regression was 67.92%. Its training score and test score are close, indicating no overfitting.



However, when we looked the feature importance, we found the most top important features are job categories and locations, which make sense because the salary must be influenced by them. However, the aim of the project is to find which skills are important for job hunters. In this case, we decided to exclude job category and location when we fitted models.

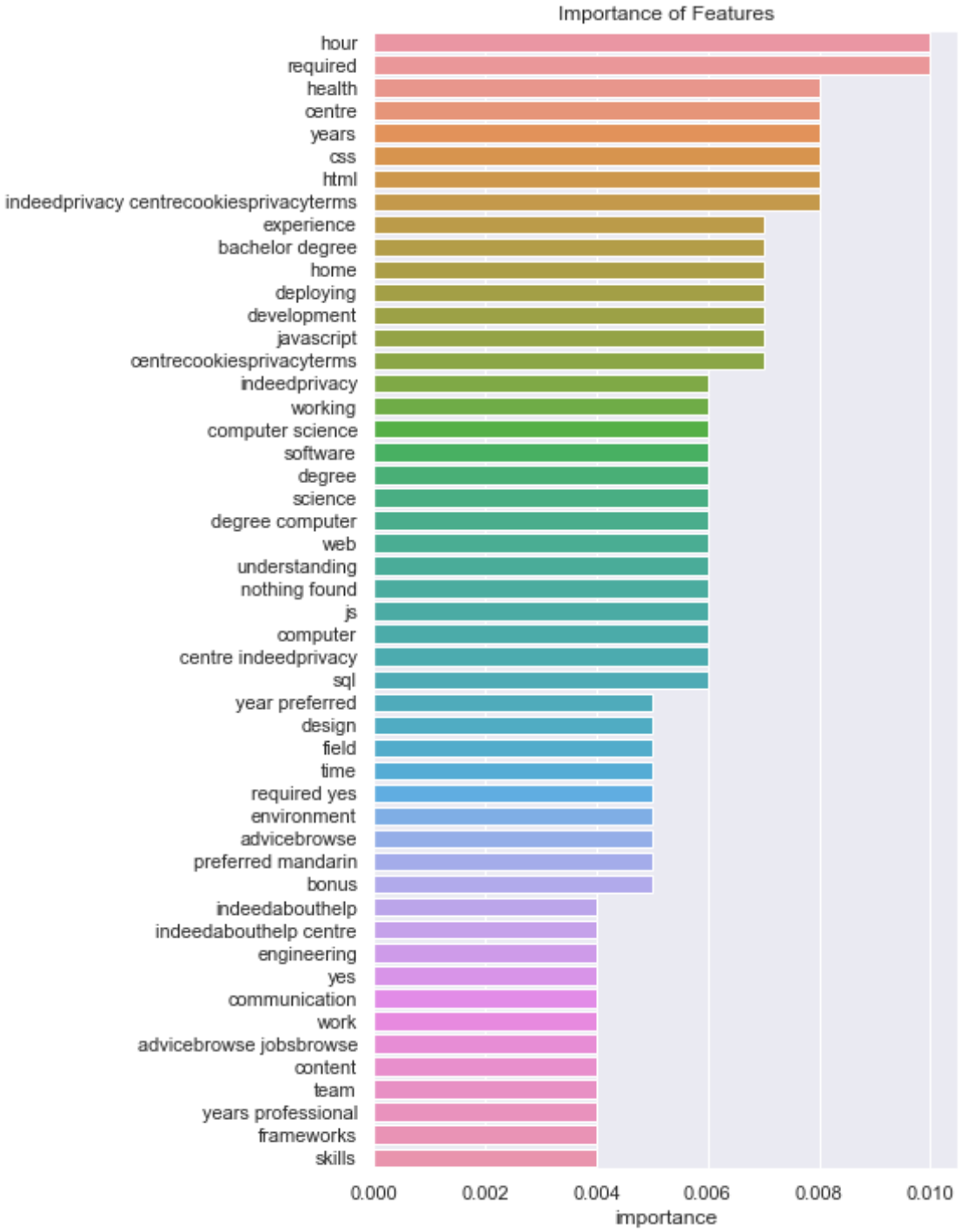
2.3.2 Model selection excluding job_category and location

```
The optimal logistic regression model:
 LogisticRegression(C=200, class_weight=None, dual=False, fit_intercept=True,
                    intercept_scaling=1, l1_ratio=None, max_iter=100,
                    multi_class='auto', n_jobs=None, penalty='l2',
                    random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                    warm_start=False)
====================================================================
The mean accuracy score is 0.654518966098511.

The optimal SVM model:
 KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                      metric_params=None, n_jobs=None, n_neighbors=20, p=2,
                      weights='distance')
====================================================================
The mean accuracy score is 0.6249661711950978.

The optimal random forest model:
 RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                        criterion='gini', max_depth=None, max_features='auto',
                        max_leaf_nodes=None, max_samples=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=50,
                        n_jobs=None, oob_score=False, random_state=44, verbose=0,
                        warm_start=False)
====================================================================
The mean accuracy score is 0.6503677770211378.
```

[37]:

	F1 score
Logistic Regression	0.654519
KNN	0.624966
Random Forest	0.650368

As shown in the table, compared to previous models, the scores of the new models changed but the logistic regression model still gives the highest score on the training set.



When we looked the feature importance again, we found the programming language like javascript, html, sql, and css are important. Besides, the degree in computer science has a significant impact on the salary as well. Moreover, the score of the logistic regression model on the test set is 65.34%.

`The weighted macro f1 score of the optimal random forest model is`

0.653350443002876 .

Finally, we checked those coefficients from the final logistic regression model. Particularly we would like to know what are the most influential features (with largest coefficients) and the least ones (with smallest coefficients).

[41]:

	0	1	2	3	4	5
10,0000-15,0000	bonus	agile scrum	scrum	knowledge	casual	design
50,000-75,000	dental	building	degree computer	facing	year preferred	django
75,000-100,000	home	related	etc experience	ability work	developmentexperience	office
<50,000	development	engineering	years	hiring	hiring labcareer	information
>150,000	years preferred	react	health	management	indeedabouthelp centrehiring	centrehiring

	6	7	8	9
10,0000-15,0000	mysql	js	communication	testing
50,000-75,000	including	ability	must	technology
75,000-100,000	technologies	process	business	project
<50,000	hour shiftmonday	software	using	writing
>150,000	centrehiring labcareer	development years	time	shiftmonday

[42]:

	0	1	2	3	4	5
10,0000-15,0000	-5.537386	-5.449267	-5.422637	-4.864413	-4.563615	-4.479798
50,000-75,000	-5.210803	-4.813210	-4.344665	-4.258374	-4.141300	-3.954722
75,000-100,000	-7.296871	-4.809430	-4.806181	-4.711514	-4.605810	-4.542210
<50,000	-3.148126	-2.824971	-2.696013	-2.489459	-2.489259	-2.356741
>150,000	-2.948756	-2.190385	-2.110509	-1.958464	-1.717015	-1.717015

	6	7	8	9
10,0000-15,0000	-4.129897	-3.854207	-3.813631	-3.718879
50,000-75,000	-3.928332	-3.902048	-3.882968	-3.801919
75,000-100,000	-4.110028	-4.063157	-3.982562	-3.667043
<50,000	-2.308185	-2.297799	-2.281447	-2.219745
>150,000	-1.717015	-1.673031	-1.624931	-1.618521

[43]:

	0	1	2	3
10,0000-15,0000	home	preferably	deploying	cloud
50,000-75,000	mandarin	web	office	required yes
75,000-100,000	bachelor degree	bachelor	bonus	en
<50,000	care	dental	must	editing
>150,000	technical	years required	bi	design

	4	5	6
10,0000-15,0000	hour	eventsdental	shiftmonday
50,000-75,000	clients	js	required hiring
75,000-100,000	development years	preferred	agile scrum
<50,000	comfortable	care hour	hour shift
>150,000	year preferred	user	capabilities

	7	8	9
10,0000-15,0000	content	identity	manage
50,000-75,000	mysql	git	experienced
75,000-100,000	experience working	travel	date
<50,000	one	space	record
>150,000	computer based	secondary	required

[44]:

	0	1	2	3	4	5
10,0000-15,0000	6.903310	5.837550	5.292294	5.235495	4.679831	4.647512
50,000-75,000	6.055229	5.514031	5.343124	5.301661	5.229200	5.194673
75,000-100,000	7.032660	5.737761	5.413230	5.318233	4.886511	4.864826
<50,000	5.511651	4.994520	4.983665	4.018811	3.840097	3.767147
>150,000	7.389142	5.965936	4.261527	3.829695	3.766348	3.706698

	6	7	8	9
10,0000-15,0000	4.557057	4.459684	4.416165	4.280178
50,000-75,000	5.070668	4.897640	4.762430	4.406164
75,000-100,000	4.857812	4.401871	4.204180	4.171346
<50,000	3.713515	3.532246	3.455834	3.292877
>150,000	3.603912	3.115978	3.035591	2.810271

2.3.3 Conclusion

We completed this project by three main steps:

(1) web scraping: To collect data, we wrote a web scraping program in Python and conduct unit tests to handle various types of information embedded in indeed website. To make scraping more effeciently, we only considered those jobs with salary feature. In total, we scraped six features for each job, including job title, job category, company name, job requirement, location and salary. Our job searching conditions ranged from 8 main cities in Canada and 5 different job titles. After that, we tidied those scattered information into a clear data frame for further processing.

(2) data visualization: Besides showing salaries distribution based on job categories and location, we utilized TF-IDF method to demonstrate popular key words within job requirements. This provided qualitative understanding of job features.

(3) model building: The core mission of this project is to predict salary based on job features. We considered three machine learning models as our candidates: logistic regression, KNN and random forest. To make overall comparison, we tried some scenarios and tune the model hyperparameters. The final evaluation showed that logistic regression outperformed other two models and achieved 65.34% on weighted F1 score. If we took a look for coefficents in final logistic regression model, it was intersting that front-end skills such as web and js had the most positive impacts on salary.

For future work, we could try scraping more data and testing other models. On the other hand, using domain knowledge to improve NLP method such as manually adding critical key words (machine learning, Python, etc..) might be another way to optimize our work.