Capstone Proposal **Pedro Felipe Gardeazábal Rodríguez**
Project: Predicting Salaries
Course: MLND - Udacity Date: 05/30/2018

Predicting Salaries from H-1B Applications

1 Definition

1.1 Project Overview

Each year potential high skilled foreign students wanting to join the US job market graduate from US universities or employers look to import missing talent. For this purpose the US H-1B visa [1] was created. It allows employers to hire high skilled foreign workers by ensuring they will be paid the current prevailing wage or higher. This ensures that employers are fair to US nationals and are not looking to import cheaper workforce. However with the job market being competitive and the prevailing wages not well known, it is important for a job seeker or an employee to determine what a fair salary is. The prevailing wage is decided by the US Department of Labor from its wage library and it is not known in advance by the future employee or employer before the application for the H-1B visa starts.

1.2 Problem Statement

When entering the job market after college for the first time or when transitioning to a new career path it is difficult to assess what the base salary for a given position should be. This is particularly true for foreign workers when moving to a new area given that salaries do not only depend on the field of knowledge, but are also dependent on the location where the future employee is going to work. In order to help not only job applicants but also employers it will be helpful to develop a model predicting the base salary a future employee should expect based on historical data collected from US H-1B visa applications.

The data set used to solve this problem contains information on location, job position and field of knowledge. This information serves as the input to the model. The data set also has the corresponding prevailing wages and employer salaries, which are continuous target variables. Therefore, the problem at hand corresponds to a supervised regression problem that can be solved by linear regression [2] or random forest regression [3].

1.3 Metrics

The purpose of this project is to develop a regression model using supervised learning. Given that the targets are continuous variables, the most appropriate metrics for evaluation correspond to those that support this kind of variable. It is important to know the difference of each prediction from the corresponding target value. In other words, the metric of interest is the absolute value of the difference between the target and the prediction. Finally, to obtain a general view of the model performance the most straightforward solution is to then calculate the average of those absolute values.

The metric of interest corresponding to the above description is the mean absolute error, in this case the mean absolute error between the targets and the predictions. The basis to judge performance is then the mean absolute error (MAE) of a model compared to the one of the benchmark model.

The mathematical expression for the MAE is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Where n is the number of data points, y_i is the target value of point i and $\hat{y}_i$ its prediction by the model.

2 Analysis

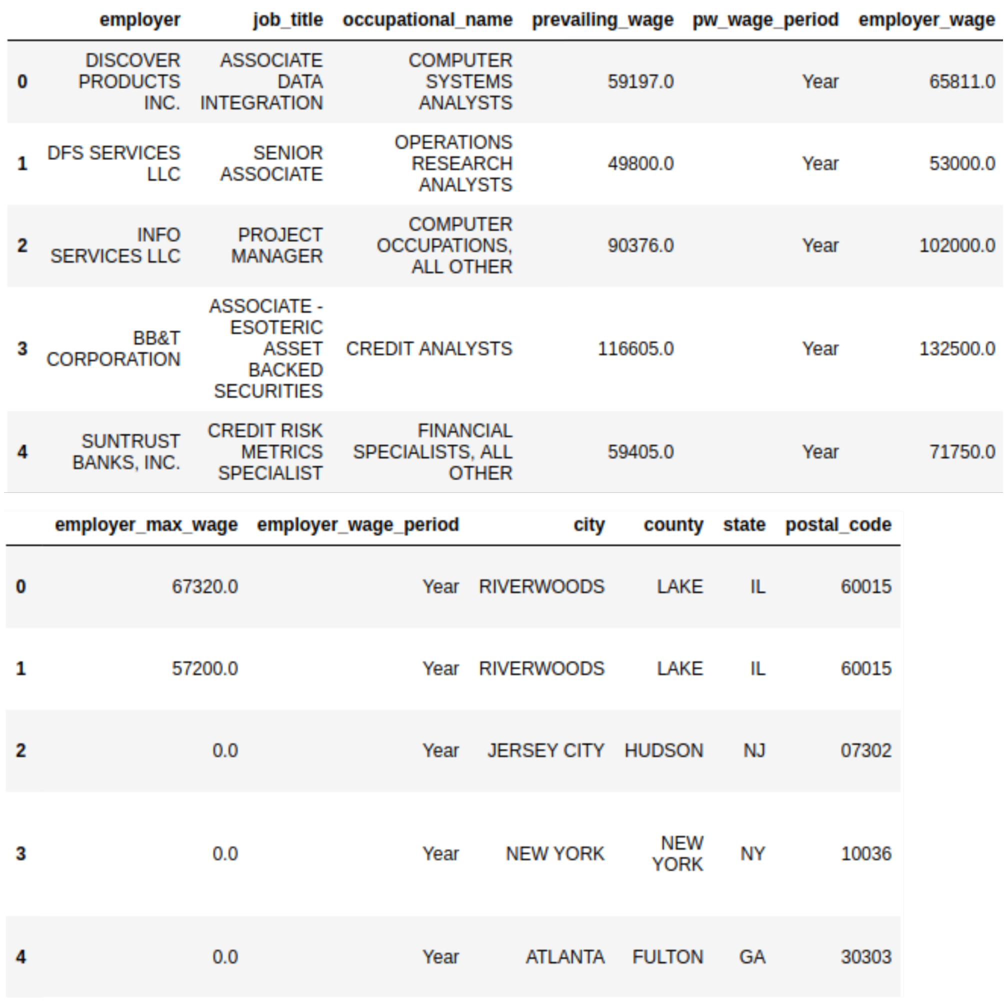
2.1 Data Exploration

For the model to be relevant in 2018 only H-1B visa applications from 2017 are considered. The data is obtained from the United States Department of Labor (USDL) and can be downloaded directly from here. From the raw file the columns of interest are:

- JOB_TITLE: Title of the job.
- SOC_NAME: Occupational name associated with the SOC_CODE.
- FULL_TIME_POSITION: Y = Full Time Position; N = Part Time Position.
- PREVAILING_WAGE: Prevailing Wage for the job being requested for temporary labor condition.
- PW_UNIT_OF_PAY: Unit of Pay. Valid values include Daily (DAI), Hourly (HR), Bi-weekly (BI), Weekly (WK), Monthly (MTH), and Yearly (YR).
- WAGE_RATE_OF_PAY_FROM: Employer's proposed wage rate.
- WAGE_UNIT_OF_PAY: Unit of pay. Valid values include Hour, Week, Bi-Weekly, Month, or Year.
- WORKSITE_CITY: City information of the foreign worker's intended area of employment.
- WORKSITE_COUNTY: County information of the foreign worker's intended area of employment.
- WORKSITE_STATE: State information of the foreign worker's intended area of employment.
- WORKSITE_POSTAL_CODE: Zip Code information of the foreign worker's intended area of employment.

To cure the data, it was defined that the model is going to predict only annual salaries. For this reason, the entries PW_UNIT_OF_PAY, WAGE_UNIT_OF_PAY and FULL_TIME_POSITION were used to express all entries from the columns PREVAILING_WAGE and WAGE_RATE_OF_PAY_FROM as annual wages. The entries in 'WORKSITE_POSTAL_CODE' came in several formats, but in order to reduce its number of categories all entries were transformed to a five digit format.

After curing, the final cleaned data set retained the following columns where the names are self-explanatory: 'employer', 'job_title', 'occupational_name', 'prevailing_wage', 'pw_wage_period', 'employer_wage', 'employer_max_wage', 'employer_wage_period', 'city', 'county', 'state' and 'postal_code'. The code to clean the data can be found at this link. A sample of the first five rows is shown below:

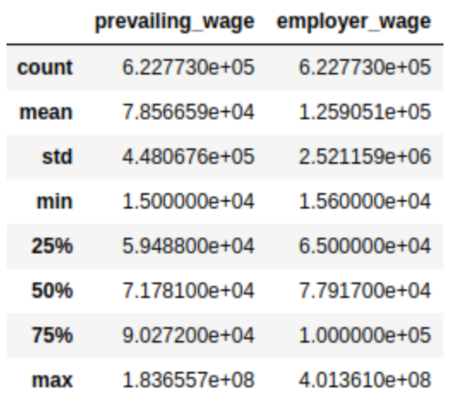


	employer	job_title	occupational_name	prevailing_wage	pw_wage_period	employer_wage	employer_max_wage	employer_wage_period	city	county	state	postal_code
0	DISCOVER PRODUCTS INC.	ASSOCIATE DATA INTEGRATION	COMPUTER SYSTEMS ANALYSTS	59197.0	Year	65811.0	67320.0	Year	RIVERWOODS	LAKE	IL	60015
1	DFS SERVICES LLC	SENIOR ASSOCIATE	OPERATIONS RESEARCH ANALYSTS	49800.0	Year	53000.0	57200.0	Year	RIVERWOODS	LAKE	IL	60015
2	INFO SERVICES LLC	PROJECT MANAGER	COMPUTER OCCUPATIONS, ALL OTHER	90376.0	Year	102000.0	0.0	Year	JERSEY CITY	HUDSON	NJ	07302
3	BB&T CORPORATION	ASSOCIATE - ESOTERIC ASSET BACKED SECURITIES	CREDIT ANALYSTS	116605.0	Year	132500.0	0.0	Year	NEW YORK	NEW YORK	NY	10036
4	SUNTRUST BANKS, INC.	CREDIT RISK METRICS SPECIALIST	FINANCIAL SPECIALISTS, ALL OTHER	59405.0	Year	71750.0	0.0	Year	ATLANTA	FULTON	GA	30303

Figure 1: First five rows of data set.

Considering that salary depends on job title and job category as well as location, the columns 'job_title', 'occupational_name', 'city', 'county', 'state' and 'postal_code' are defined as the inputs. All this variables are categorical and need a special treatment when developing the models. Their treatment is described in section 3.1.

On the other hand the targets, 'prevailing_wage' and 'employer_wage', are numerical variables. As figure 2 shows, their values are spread in a long range going from around \$15,000 to more than \$100,000,000 while their mean value remains low compared to their maximum value. Their third quartile also remains low when compared to the maximum value. All this is an indication of the



	prevailing_wage	employer_wage
count	6.227730e+05	6.227730e+05
mean	7.856659e+04	1.259051e+05
std	4.480676e+05	2.521159e+06
min	1.500000e+04	1.560000e+04
25%	5.948800e+04	6.500000e+04
50%	7.178100e+04	7.791700e+04
75%	9.027200e+04	1.000000e+05
max	1.836557e+08	4.013610e+08

Figure 2: Target variables statistics.

presence of several outliers that need to be removed before start developing the model.

The outliers can be removed without fear of losing to much information due to the size of the data set: the total number of data points is 622,773.

2.2 Exploratory Visualization

As noted in section 1.1, employer salary must be equal or higher than prevailing wage per USDL guidances, so it appears that prevailing wage may be an important predictor of employer wage.

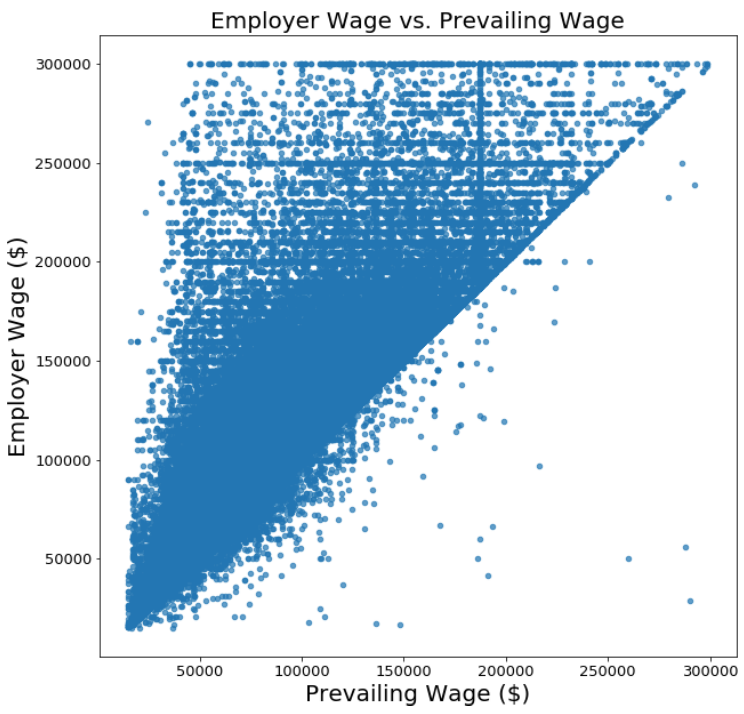


Figure 3: Employer wage as a function of prevailing wage.

In order to confirm this hypothesis let's see how employer wage compares to prevailing wage by looking at the plot in figure 3.

As expected, most of the data points fall above a diagonal line confirming that most of the employer wages are higher or equal than their respective prevailing wage. However there are some points not following this rule and that need to have a special treatment in the preprocessing step (section 3.1). From the plot it is difficult to state how good prevailing wage is as predictor for employer wage, however the Pearson correlation coefficient (PCC) between these two variables gives us a better picture. The PCC, when the values are smaller than $300K, has a value of 0.8633, this indicates that the prevailing wage is highly correlated to employer wage when outliers are removed and then can be used as a good predictor.

2.3 Algorithms and Techniques

Since the purpose of this project is to predict an expected salary based on historical data, supervised learning using regression algorithms is used. The location, job title, and prevailing wage are used as predictors in trying to ascertain the target salary. According to USDL guidelines wages have to be equal or higher than the prevailing wage. It is fair to assume that a user of the developed model will not know the prevailing wage in advance, so to train the model, after dividing the data into training and validation sets, the training data is further divided in two. One set is used to predict prevailing wages as a function of location, job title and work category, and the second set is used to predict the final target salary as a function of the predicted prevailing wage and the other employment information. Each of these subsets is further divided in two in order to have training and testing sets to develop the models.

All the algorithms used in this project to develop the models are from the scikit-learn library. Among the linear regression algorithms in that library [4], the following are used The linear regression algorithms to consider are:

- Linear regression [5]:

  It corresponds to the Ordinary Least Squares method in which the coefficients resulting from the regression correspond to those minimizing the residual sum of squares. In other words, considering the predictions, the function to minimize is:

  $$RSE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i|^2$$

  Where n is the number of data points, y_i is the target value for point i and $\hat{y}_i$ its prediction by the linear regression.

- Huber regression [6]:

  It is a regression model that uses a loss function optimized for outliers. This is well suited for our data because in the data analysis it was shown the presence of this type of points. The function a Huber regressor minimizes is:

  $$\sum_{i=1}^{n} \left[\sigma + H_m\left(\frac{y_i - \hat{y}_i}{\sigma}\right)\sigma \right] + \alpha ||w||^2$$

where

$$H_m(z) = \begin{cases} z^2 & \text{if } |z| < \epsilon \\ 2\epsilon|z| - \epsilon^2 & \text{if } |z| \geq \epsilon \end{cases}$$

Here σ is a parameter that the algorithm optimizes as it does with the squared norm of the weights $||w||^2$. α and ϵ are hyper-parameters that are adjust by a grid-search cross-validation.

- SGD regression [7]:

  SGD stands for stochastic gradient descent and it is a model that adapts very well to problems with a very large set of data points. In the case of large data sets this model can be trained by batches. In the present case it was found that the hyper-parameter that affects the most the output result is the number of iterations. The remaining hyper-parameters are kept at their default values and are not further explored in detail due to the long training time.

In addition to linear regression, among the ensemble methods available [8], random forest [9] is explored. Ensemble methods correspond to the combination of the predictions from several estimators built from one learning algorithm. This allows for better generalization and robustness.

In the case of random forest each estimator corresponds to a decision tree. When building one decision tree only a subsample of the data set is taken. The subsample points are chosen randomly by the random forest algorithm and the decision tree is optimized only for those points. When a random forest model makes a prediction, the output consists on the average prediction of the single decision trees.

Random forest in explored among the regression algorithms because all the predictors correspond to categorical variables that are well handed by decision trees.

Benchmark Model

As stated in the previous section, the solution for the proposed project has two important steps: to predict prevailing wage and estimate employer wage. For the first part a standard linear regression is used as the benchmark model. The target being salaries, it is important to estimate in average how far the predictions are from the targets. This average is the basis to evaluate how accurate the developed models are compared to the benchmark model.

How to find the hyper-parameters for the benchmark model of the first step is explained in section 3.2. Its MAE, which is the metric of choice to evaluate the models, is \$10,534.496.

For the second part of the problem (predicting employer wage) the benchmark model is a SGD regressor [7] using the prevailing wage from the data set, while the models to be evaluated use the predicted prevailing wage from the model obtained in the first part or will not consider prevailing wage at all. Similarly in this step the metric of interest is how far on average the predictions are from the targets. How to train the SGD regressor benchmark model is explained in section 3.2. For the moment, it is relevant to mention that its MAE is \$9,671.643.

3 Methodology

3.1 Data Preprocessing

The only numerical columns in the data set are the targets, 'prevailing_wage' and 'employer_wage'. As mentioned earlier, 'prevailing_wage' is supposed to be equal or smaller than 'employer_wage', however this is not the case for all the columns in the data set. For the rows were this condition is not respected, it is assumed that the person filling the corresponding H-1B form made a mistake by swapping the fields. So, for those columns in the data set 'prevailing_wage' and 'employer_wage' are swapped back correctly using the simple code below:

```
idx = data.prevailing_wage > data.employer_wage
data.loc[idx, ['prevailing_wage', 'employer_wage']] \
                        = data.loc[idx, ['employer_wage', 'prevailing_wage']].values
```

After this process, the plot below confirms that now all the points follow the condition set by USLD:

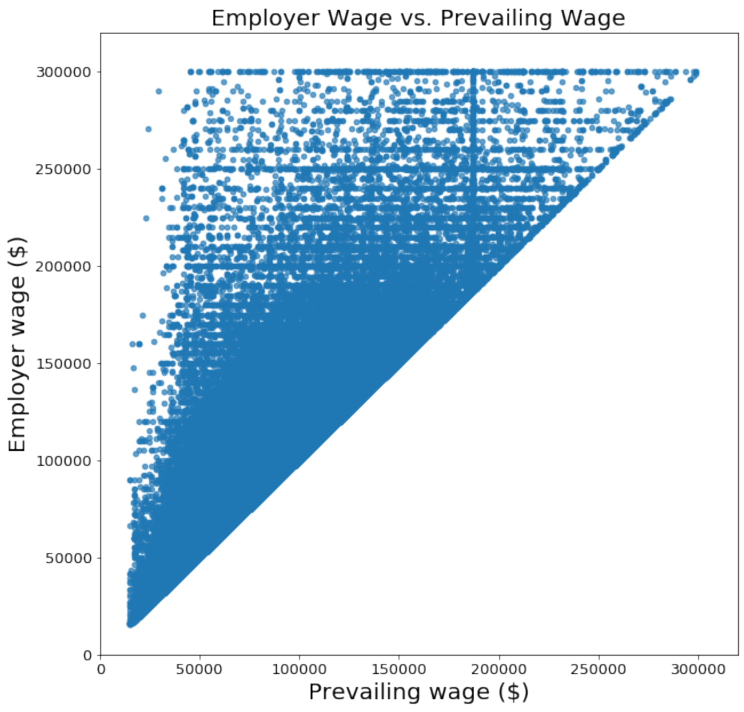


Figure 4: Employer wage as a function of prevailing wage. Plot done after applying the code to swap 'prevailing_wage' and 'employer_wage' when 'prevailing_wage' > 'employer_wage'

The PCC between 'prevailing_wage' and 'employer_wage' is now 0.8641, which confirms that 'prevailing_wage' is still a good predictor for 'employer_wage'.

To remove outliers Tukey's method [10] is applied to the numerical columns 'prevailing_wage' and 'employer_wage'. Tukey's method removes points having a value smaller than the 1st quartile minus 1.5 times the interquartile range, and points higher than the 3rd quartile plus 1.5 times the interquartile range. After this process the number of points left is 587,905 and the PCC between 'prevailing_wage' and 'employer_wage' slightly increases to 0.8721.

In the data set all columns are categorical except for 'prevailing_wage' and 'employer_wage', the former being the target variable and the latter ignored in this first part. Linear models necessitate numerical inputs but the categorical variables of the data set correspond to text and some of them are short sentences. The most direct approach to obtain numerical inputs is to concatenate per row the values of these columns into a single string by keeping spaces between words. The new created column is called 'merged'. The 'merged' variable is then encoded using a TF-IDF vectorizer that allows its expression in a matrix form per data point so it can be used for a numerical linear regression. To choose the right parameter for the n_gram_range hyper-parameter of the TF-IDF vectorizer a grid-search cross-validation is used when defining the benchmark model. This step shows that the proper value for it is its default setting, meaning that not n-gram is necessary.

Scikit-learn doesn't handle text as inputs for categorical variables. So, to use the random forest algorithm all the categorical columns in the data set must be numerically encoded. For this, it is enough to count the number of different labels per column and assign a number to each label. However this kind of numerical encoding can be interpreted as giving more importance to labels with higher assigned number. This can introduce a bias to the algorithm. To deal with this issue the most common labels in a category are assigned the higher values.

This process is performed first by obtaining the number of counts per each category to create a dictionary for each column where the most common categories are assigned higher numbers:

```
# Dictionary containing mapping codes for each feature
coding_dic1 = {}
for column in pr_X.columns:
    # Counting values
    map_counts = pr_X_train[column].value_counts()
    # Assign maximum value to most common label
    coding_dic1[column] = \
        dict(zip(map_counts.index, np.arange(map_counts.shape[0] - 1, -1, -1)))
```

Then a function is created to add to the data set new columns where categorical variables are encoded using the created dictionaries:

```
def encode_feats(df, coding_dic):
    feats_encoded = []
    for feat in coding_dic.keys():
        feat_dict = coding_dic[feat]
        df[feat+'_enc'] = df[feat].map(feat_dict)
        feats_encoded.append(feat+'_enc')
    return feats_encoded
# Apply encode_feats to training data
feats_enc = encode_feats(pr_X_train, coding_dic1)
pr_X_train.head()
# Apply encode_feats to test data
feats_enc = encode_feats(pr_X_test, coding_dic1)
pr_X_test.fillna(-1, inplace=True)
```

As shown above, the encoding is done in the training set and then applied to the test set. Not all the labels in the test set have a corresponding label in the training set. These labels are filled as null values when encoding the columns, so they are replaced with a -1. This value is properly recognized by the random forest as a label not present in the category.

Finally, since the targets spread in a large range from 20,000 to more than 300,000, their natural logarithm is taken to train the models. Using the natural logarithm considerably reduces the range of the targets and allows for an easy transformation to recover back the proper predicted values.

3.2 Implementation and Refinement

To calculate the MAE for each model the function mean_absolute_error [11] from the module sklearn.metrics is used. 20% of the data is kept as a validation set and the rest is used for training. However, given that two predictions need to be performed the training set is further divided in half. The first half is used to develop the model predicting prevailing wages and is identified by the prefix 'pr_'. The second half is used for the model to predict employer wages and is identified by the prefix 'em_'. These two set are further divided into training and test sets (20% is used for testing).

The first step is to perform a linear regression for the benchmark model. In this step it is also determined if the most appropriate TF-IDF vectorizer is a simple one or one corresponding to an n-gram using a 5-fold cross-validation grid-search with 'neg_mean_absolute_error' as the scoring function. In this step a pipeline is used to first fit the TF-IDF vectorizer and then fit the linear regression using the output from the vectorizer:

```python
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LinearRegression
from sklearn.pipeline import Pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from time import time

# Definition of random seed
seed = 14
# Define pipeline and steps
steps = [('vect', TfidfVectorizer(sublinear_tf=True)), # Word vectorizer
         ('linr', LinearRegression())]
pipeline = Pipeline(steps)
# Parameters for grid search
parameters = {'vect__ngram_range': [(1, 1), (1,2), (1,3)]}
# Defining cross validation object
cv = KFold(n_splits=5, shuffle=True, random_state=seed)
# Define grid search
grid_search = GridSearchCV(pipeline, param_grid=parameters, \
                           cv=cv, scoring='neg_mean_absolute_error')
# Train
start = time()
grid_search.fit(pr_X_train.merged, pr_y_train)
end = time()
```

The code above shows that an n-gram is not necessary in the TF-IDF for the benchmark model. Training over the whole training data available results, after testing, in an MAE of $10,534.496.

After developing the benchmark model, a Huber regressor [6] which is less sensitive to outliers is considered by using the same input as the benchmark model. Again a 5-fold grid-search cross-validation is performed to find the best hyper-parameters:

```
from sklearn.linear_model import HuberRegressor

# Define pipeline
steps1 = [('vect1', TfidfVectorizer(sublinear_tf=True)), # Word vectorizer
         ('hubr', HuberRegressor(max_iter=600))]
pipeline1 = Pipeline(steps1)
# Grid search parameters
parameters = {'hubr__epsilon': [1, 1.05, 1.15, 1.25],
              'hubr__alpha': [0, 0.1, 0.01]}
# Grid search
grid_search = GridSearchCV(pipeline1, param_grid=parameters, \
                           cv=cv, scoring='neg_mean_absolute_error')
# Train
start = time()
grid_search.fit(pr_X_train.merged, pr_y_train)
end = time()
```

The optimum values for the epsilon and alpha parameters are found to be 0.1 and 1 respectively. The best maximum number of iterations, max_iter, is found manually to be 600. After training over the whole training data with the proper parameters and testing on the test set, the MAE is found to be $10,263.151, which is a slight improvement over the benchmark model.

As an additional model a random forest regressor is used. For this model the input is no longer the 'merged' column but instead the encoded columns defined in section 3.1 are used. All the parameters were kept as default except for the number of estimators. The appropriate numbers of estimators is found manually to be 100:

```
from sklearn.ensemble import RandomForestRegressor
# Training
forest = RandomForestRegressor(n_estimators=100, random_state=seed, n_jobs=20)
start = time()
forest.fit(pr_X_train[feats_enc], pr_y_train)
end = time()
```

This regressor over the test set resulted in an MAE of $9,376.254, which is superior to the previous models.

For the second part, the initial model is used to predict prevailing wages. These prevailing wages combined with the columns 'state' and 'postal_code' are used as inputs for the final model. The target salaries are employer wage. As in the first part, all the values are expressed in natural logarithm form. In this part, the benchmark model is an SGD regressor. For the benchmark model the prevailing wage from the data set is used. However, as mentioned before, it is fair to assume that a user of the final model will not know the prevailing wage, so for the other models the predicted prevailing wages from the initial model are used. Since the amount of data and categories are too big to handle by an SGD regressor, the training is done in batches using the OneHotEncoder method from the dummyPy package [12]:

```python
from dummyPy import OneHotEncoder

# Features to use
features1=['state', 'postal_code', 'pw_log']
# Create one-hot encoder and fit it to the data
encoder = OneHotEncoder(['postal_code', 'state'])
encoder.fit(em_X_train[features1])
```

To train by batches the SGD model a function creating an iterator needs to be defined first:

```python
def batches(size, X, y):
    # This function transform data into batches
    batch_start = 0
    n_points = X.shape[0]
    while batch_start < n_points:
        batch_end = batch_start + size
        if batch_end < n_points:
            X_batch = X.iloc[range(batch_start, batch_end)]
            y_batch = y.iloc[range(batch_start, batch_end)]
        else:
            X_batch = X.iloc[range(batch_start,n_points)]
            y_batch = y.iloc[range(batch_start, n_points)]
        yield X_batch, y_batch
        batch_start += size
```

Afterwards the SGD regressor is trained using its partial_fit method:

```python
from sklearn.linear_model import SGDRegressor

# Set parameters for batch training
batch_size = 20000
# Create regressor
regressor1 = SGDRegressor(max_iter=600, random_state=seed)
# Get batches
train_batches = batches(batch_size, em_X_train[features1], em_y_train)

# Start training
start = time() #start time
for X_batch, y_batch in train_batches:
    # Transform batch to one-hot encoding
    X_batch_trans = encoder.transform(X_batch)
    # Partial fit
    regressor1.partial_fit(X_batch_trans, y_batch)
end = time() #end time
```

Similarly, the predictions over the test set is done by batches and results in an MAE of $9,671.643.

This is the benchmark model for the second part.

To find the best model in the second part, the prevailing wage as predicted by the model from the first part is used and the prevailing wage from the data set is ignored. The straight option is to train again the SGD regressor. This model results in an MAE of $12,414.213. For comparison, the SGD regressor is also trained without accounting for prevailing wage, so only the columns 'state' and 'postal_code' are used. This results in an MAE of $17,556.142, confirming that prevailing wage is an important predictor if only 'state' and 'postal_code' are used.

Similar to the first part, in addition to the SGD regressor, a Huber and random forest regressors are developed and compared. The Huber regressor is trained on;y over the 'merged' column (see section 3.1) and results on an MAE of $11,936.616.

For the random forest regressor the encoded columns are created using only the data available in this step. So the dictionaries used to create the new columns must be created again similarly as it was done in the first part. Afterwards, the random forest regressor is trained without predicted prevailing wages and with predicted prevailing wages. Each of these regressors resulted on an MAE of $11,297.345 and $10'473.161 respectively.

4 Results

4.1 Model Evaluation and Validation

To choose the final model two decisions need to be made: chose the model to predict prevailing wage and chose the model to predict employer wage. The model predicting prevailing wage is referred as model 1 and the model predicting employer wage is called model 2. The final model is then composed by model 1 and model 2.

The decision on which regressors are kept for model 1 and 2 is made by comparing MAEs. Give that most of the models developed have an MAE higher than $10,000, inspired by ensemble methods, a combination of the best regressors for model 1 and model 2 are also evaluated.

Model 1 the combined model consists of the Huber and random forest regressors. For model 2 the combined model is produced using the Huber regressor and the and the random forest regressor that uses as predictor the prevailing wage predicted from model 1. The combinations are a simple average of the predictions by each individual model. Table 1 summarizes the results for model 1 and model 2.

The models with the smallest errors, different for the benchmark model, are chosen to form the final model. The green rows in the table identify the best choices for model 1 and model 2. For model 1 the best regressor is the combination of the Huber and random forest regressors. For model 2 the best model corresponds to the random forest regressor that considers the prevailing wage predicted by model 1. In model 2 the benchmark model cannot be considered because it uses the prevailing wage from the data but this is information unknown to a future user of the final model.

4.2 Justification

The final model, which is formed by model 1 and 2, is evaluated on the initial validation set. Remember that the final goal is to predict employer wage but to predict employer wage, prevailing wage must be predicted first.

Model 1: Predict Prevailing Wage	
Model	MAE
Benchmark Model Linear Regression	$10,534.496
Huber Regressor (A)	$10,263.151
Random Forest (B)	$9,376.254
Combination (A and B)	$9189.375

Model 2: Predict Employer Wage	
Model	MAE
Benchmark Model SGD Regressor Prevailing Wage Data	$9,671.643
SGD Regressor Prevailing Wage Predicted From Combined Model 1	$12,414.123
Huber Regressor (C) Without Prevailing Wage	$11,936.616
Random Forest Without Prevailing Wage	$11,297.345
Random Forest (D) Prevailing Wage Predicted From Combined Model 1	$10,473.161
Combination (C and D)	$10,636.065

Table 1: Mean absolute error for tested models.

The validation set consists of 20% of the data after outliers removal. Even if the final interest is not about prevailing wage, it is interesting to note that in this set the MAE results in a value of $9,140.855. This is slightly better but almost similar to the MAE of model 1 in its testing set. This is an indication that model 1 generalizes well to unseen data.

Considering the employer wage, which is the purpose of this project, the MAE in the validation data by model 2, and hence by the final model, gives a value of $10,389.571. It is worth to note that as with model 1, this value is also slightly better but almost similar to the MAE of model 2. So it can be said that the final model generalizes well to unseen data.

The final MAE is better than the one reported during model training. And this is true for both prevailing wage and employer wage. So the final model generalizes very well to unseen data.

5 Conclusion

A final model consisting of two models was proposed in order to predict a salary as a function of location, job title and position. Given that the data base with the target salaries corresponds 2018 H-1B applications, it was chosen to develop the final model in two steps. According to USDL guidelines a salary for a foreign worker must be higher or equal than the area's prevailing wage for a given position. The first step was to develop a model predicting prevailing wage to then develop a model predicting salary.

Given that the target salaries were presented in the data set, both models were based on supervised learning using regression algorithms. The algorithms of choice corresponded to linear regressions and random forest. In both cases the benchmark model was based in a simple linear regression. To improve the benchmark models more advanced versions of linear regression were considered such as Huber and SGD regressors. The Huber regressor was chosen because it is well adapted to data with the presence of strong outliers. The choice of the SGD regressor was based on the large size of the data set and the high number of categories for each categorical variable. The partial_fit method of the SGD regressor adapts very well to data sets with those characteristics.

Additionally, in the search for generalization and robust models, a random forest regressor was considered. It was found that this kind of model tends to generalize very well even when the unseen data contains categories the regressor has never encountered during training. The resulting random regressor models were superior to the linear models. However inspired by ensemble methods the combination of random forest with linear regression was explored. The combination consisted on the simple average of the predictions by each model. This resulted in a better model for the first part of predicting prevailing wage but it didn't make a difference for the final model of predicting employer wage.

The metric of choice to evaluate the models as well as the final model was the mean absolute error (MAE). The MAE found in the validation set was of around $10,400. This value may seem high but we should remember that the output of the final model is to produce a base to negotiate salary as a newcomer. Figure 5 shows how the predicted salaries compare to their targets. It shows that as the target salary increases the model tends to underestimate the predicted salary. In that sens the final model doesn't generalizes as well as expected but at least most of the predicted salaries are higher than the predicted prevailing wage and hence is proposed that it is necessary to consider both, the predicted prevailing wage and employer wage to make a more informed final decision.

The above remark suggests that the there is information the model was not able to capture in order to perform proper predictions. This is particularly true for high salaries. This means that further domain knowledge needs to be applied to improve the models. A better domain knowledge allows for the creation of better features or predictors. For example given that it is considered that

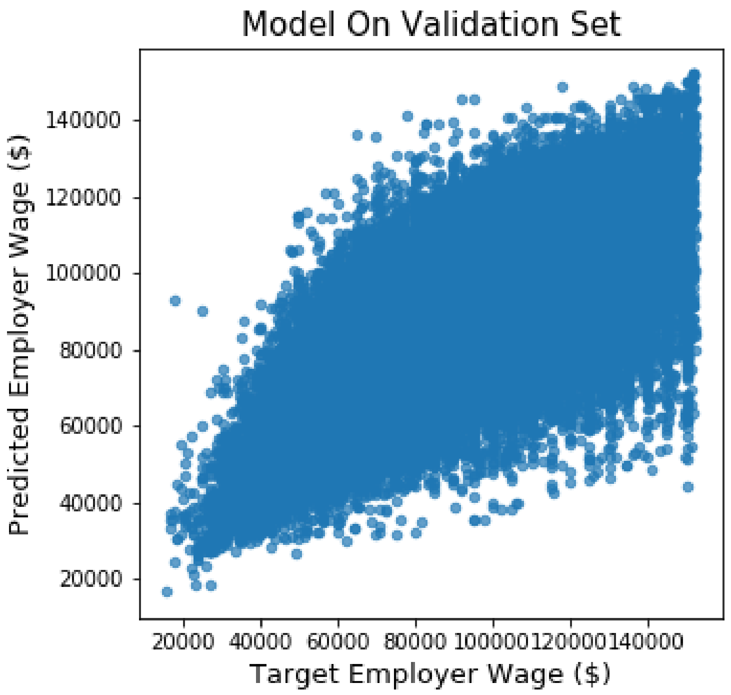


Figure 5: Comparison between target wage salary and predicted wage salary on validation set.

when qualified labor is rare in an area salaries tend to be high in order to attract talent, in this case we may consider as a feature the total number of applications per state or city.

Finally, an alternative way to improve the solution to the proposed problem is to take advantage of the remarks about the random forest regressor applied to this data set. It was found that random forest not only gave the better metrics but it also generalized very well to unseen data. So it would be interesting to see what happens if we turn this problem instead into a supervised classification learning one. Under this scenario the target salaries can be expressed as categories with each category corresponding to a salary a range. The advantage of this approach is that the predicted probabilities for the different target categories can be compared among them, hence giving a better insight into the final results.

References

[1] United States Department of Labor, Wage and Hour Division (WHD), H-1B Program. https://www.dol.gov/whd/immigration/h1b.htm

[2] Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani. Chapter 3, *An Introduction to Statistical Learning*. 8th Printing. Springer, 2017. 59-104. Print.

[3] Section 8.8.2. *Ibid.*, 319-323.

[4] http://scikit-learn.org/stable/modules/linear_model.html

[5] http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[6] http://scikit-learn.org/stable/modules/linear_model.html#huber-regression

[7] http://scikit-learn.org/stable/modules/sgd.html#regression

[8] http://scikit-learn.org/stable/modules/ensemble.html

[9] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[10] http://datapigtechnologies.com/blog/index.php/highlighting-outliers-in-your-data-with-the-tukey-method/

[11] http://scikit-learn.org/stable/modules/generated/sklearn.metrics.mean_absolute_error.html

[12] Yashu Seth. A BLOG ON DATA SCIENCE, MACHINE LEARNING AND ARTIFICIAL INTELLIGENCE. *How to One Hot Encode Categorical Variables of a Large Dataset in Python?*. 2017.