PHASE 3: MODELLING AND ERROR ANALYSIS

1. MODELS WITH BOW FEATURES

I have created the input and output data by combining extracted features and bow representations of resumes and job descriptions.

- We have 90 data points.
- We have 1444 features.

1.1. Train Test split

Outputs of both test and train data have the same distribution.

At first, I had divided the data into train, cv, and test data with 60%, 20%, and 20% respectively in each group.

But the models were not showing very different results when I changed the random state for the split.

This is because we have a very small dataset.

So I decided to use K-fold cross-validation for hyperparameter tuning. And I have divided the data as follows,

- Train data is 70% and has 63 data points.
- Test data is 30% and has 27 data points.

1.2. Preprocessing

In preprocessing I have done standardization which means mean centering and variance scaling. So now all columns will have mean of 0 and variance of 1.

1.3. Modeling (Basic)

Initially, I have tried only two models. One is Linear Regression with L2 regularizer which is a simple model. And the other is KNN Regression which is a little bit of complex model.

After hyper-param tuning with hyperopt, I found the best alpha for the linear regression model is 0.109. But when we calculate the MSE errors we found that there is huge gap in train and test mse errors. The mse on train data is 4.79 whereas the mse on test data is 91.48.

So, I plotted the test and train mse loss for different alpha values. From the plot, we can see that the model is overfitting. The difference between train and test mse decreases for higher alpha values. But in that case, both mse losses will increase significantly.

Since Linear regression is a simple and high bias model we should not see such overfitting. I had also tried linear regression with l1 regularizer and it was showing a very similar results.

Similarly for KNN regression, I found the best params (n_neighbors=7, metric='cosine'). But the KNN is performing very badly on both train and test data. The mse on train is 157.02 and test is 148.5.

So I chose the Linear regression model with l2 regularizer for error analysis.

1.4. Error Analysis

For BoW, I had used a minimum document frequency of 2, but the models were overfitting so I chose min_df = 4. Which improved the result a little bit but not much.

I checked the data points which showed higher errors and compared different resumes.

I also tried the linear regression model multiple times and checked the feature importance. It showed some traces of collinearity.

These collinearities or multi-collinearities could be because we do have short and long forms of some skills and degrees. And also because we have unigram, bigram, and trigram.

So I did forward feature selection with SequentialFeatureSelector to select 500 features. After that when I fitted a linear regression. And I found that only less than 100 weights were non-zero.

So again did forward feature selection to select only the top 100 features. So now we have reached from 1444 features to 100 features, which means we have only 6.93% of the original features.

When we view the selected features we see that we have some of the extracted features. Also, we have more number of features or words from the resume than the jd. That's because we have only one jd for the whole dataset.

1.5. Train Test split and Preprocessing

Now again I have done train test split with the same random state. So both the data will contain the same data point as earlier. I have done the mean centering and variance scaling. And also I have saved the data points.

1.6. Modeling

Now again I have done hyper-param tuning for L2 Linear Regression.

- The best alpha is 1.659.
- The train mse and r-squared are 1.9844 and 0.9924 respectively.
- The test mse and r-squared are 52.1386 and 0.7784 respectively.

For KNN Regression,

- The best params are n_neighbors=7 and metric='cosine'
- The train mse and r-squared are 130.7427 and 0.5014 respectively.
- The test mse and r-squared are 113.766 and 0.5166 respectively.

For Decision Tree Regressor,

- The best params are max_depth=5, min_samples_split=2
- The train mse and r-squared are 21.9602 and 0.9163 respectively.
- The test mse and r-squared are 164.5762 and 0.3006 respectively.

For Support Vector Regression (RBF Kernel),

- The best param is C = 253.341
- The train mse and r-squared are 0.01 and 1 respectively.
- The test mse and r-squared are 90.842 and 0.614 respectively.

For Support Vector Regression (Linear Kernel),

- The best param is C = 1.429
- The train mse and r-squared are 3.0771 and 0.9883 respectively.
- The test mse and r-squared are 49.5258 and 0.7895 respectively.

For Random Forest Regressor,

- The best params are n_estimators = 50, min_samples_split = 5, max_depth = 15
- The train mse and r-squared are 27.0421 and 0.8969 respectively.
- The test mse and r-squared are 125.082 and 0.4685 respectively.

For XGBoost Regressor,

- The best params are n_estimators = 150, max_depth = 2,learning_rate = 0.7, reg_lambda = 38.924

- The train mse and r-squared are 3.2299 and 0.9877 respectively.
- The test mse and r-squared are 133.6595 and 0.432 respectively.

Here we can see that the best model is Linear kernel based SVR followed by the Linear regression with l2 regularizer.

2. MODELS WITH AVERAGEWORD2VEC FEATURES

I have created the input and output data by combining extracted features and average word2vec representations of resumes and job descriptions.

- We have 90 data points.
- We have 612 features.

2.1. Train Test split

Again we have done a train-test split just like before. Since we have used the same random state. The data points will be same as before in train and test splits.

2.2. Preprocessing

Again I have done column standardization with mean centering and variance scaling.

2.3. Modeling (Basic)

I have used Linear Regression with l2 regularizer and KNN Regression for initial analysis.

After hyperparameter tuning for Linear Regression, I found the best alpha=88.674.

And train and test mse are 17.155 and 111.87 respectively.

For KNN Regression the best hyperparameters are n_neighbors=7 and metric='cosine.

And train and test mse are 155.137 and 100.862 respectively.

We can see the results are not very promising.

2.4. Error Analysis

From the previous experience and from the fact that we have only one jd, I had a hunch that we can get better results after doing forward feature selection.

I used the linear regression for forward feature selection and selected 200 features out of 613 features.

2.5. Train Test split and Preprocessing

Just like before, we have done train test split. Again we have preprocessed the data to have mean of 0 and variance of 1. And save them in file.

2.6. Modeling

For L2 Linear Regression.

- The best alpha is 8.287.
- The train mse and r-squared are 4.4046 and 0.9832 respectively.
- The test mse and r-squared are 59.5001 and 0.7472 respectively.

For KNN Regression,

- The best params are n_neighbors=7, metric='cosine'
- The train mse and r-squared are 105.3122 and 0.5984 respectively.
- The test mse and r-squared are 83.3849 and 0.6457 respectively.

For Decision Tree Regressor,

- The best params are max_depth=10, min_samples_split=5
- The train mse and r-squared are 2.4671 and 0.9906 respectively.
- The test mse and r-squared are 389.3863 and -0.6546 respectively.

For Support Vector Regression (RBF Kernel),

- The best param is C = 399.718
- The train mse and r-squared are 0.01 and 1 respectively.
- The test mse and r-squared are 53.8473 and 0.7712 respectively.

For Support Vector Regression (Linear Kernel),

- The best param is C = 385.101
- The train mse and r-squared are 0.01 and 1 respectively.
- The test mse and r-squared are 109.3227 and 0.5355 respectively.

For Random Forest Regressor,

- The best params are n_estimators = 50, min_samples_split = 5, max_depth = 20
- The train mse and r-squared are 26.4767 and 0.899 respectively.
- The test mse and r-squared are 113.132 and 0.5193 respectively.

For XGBoost Regressor,

- The best params are n_estimators = 100, max_depth = 2,learning_rate = 0.04, reg_lambda = 0.22
- The train mse and r-squared are 9.3477 and 0.9644 respectively.
- The test mse and r-squared are 109.5243 and 0.5346 respectively.

The RBF kernel based SVR have the lowest mse for test data. But when we compare train and test mse for it then it is clear that the model is overfitting a lot.

The Linear regression will be the best choice among these models. Since it has lower mse and also it is not overfitting that much.

3. PERFORMANCES

The top 3 models based on MSE on test data are,

1. Support Vector Regression (Linear Kernel) model based on BoW feature
2. Linear Regression (L2 Regularizer) model based on BoW feature
3. Support Vector Regression (RBF Kernel) model based on Average Word2Vec feature

The worst performing models are Decision Tree Regressor models.

4. STACKING ENSEMBLE (STACK OF BEST MODELS)

I was not much impressed with the result. So I thought, can we somehow combine the best of both the features to create a better model.

That reminded me of the stacking ensemble. So I chose the following models for level-0 of stacking,

- Support Vector Regression (Linear Kernel) model based on BoW features
- Linear Regression (L2 Regularized) model based on Word2Vec features

4.1. Preprocessing

I have done similar preprocessing as above. I have done mean centering and variance scaling.

4.2. Finding the Meta model

The stacking gave very good results. Here's what I found for stacking ensemble models.

For L2 Linear Regression.

- The best alpha is 0.068.
- The train mse and r-squared are 2.1757 and 0.9917 respectively.
- The test mse and r-squared are 33.8341 and 0.8562 respectively.

For KNN Regression,

- The best params are n_neighbors=2, metric='euclidean'
- The train mse and r-squared are 1.5009 and 0.9943 respectively.
- The test mse and r-squared are 20.6566 and 0.9122 respectively.

For Support Vector Regression (RBF Kernel),

- The best param is C = 271.031
- The train mse and r-squared are 2.0037 and 0.9924 respectively.
- The test mse and r-squared are 30.6849 and 0.8696 respectively.

For Support Vector Regression (Linear Kernel),

- The best param is C = 8.179
- The train mse and r-squared are 2.661 and 0.9899 respectively.
- The test mse and r-squared are 44.1259 and 0.8125 respectively.

The KNN regressor seems to be performing the best as the meta-model for the stacking regressor.

4.3. Train and Test performance with the best model

With our best model which is the stacking ensemble model, where the base models or level-0 models are the *Support Vector Regression (Linear Kernel) model based on BoW features* and *Linear Regression (L2 Regularized) model based on Word2Vec features*, and the meta-model or level-1 model is a *KNN Regressor*.

We get the following results,

- Train MSE is 1.5009 and Test MSE is 20.6566
- Train R-Squared is 0.9943 and Test R-Squared is 0.9122
- Train MAE is 0.8253 and Test MAE is 3.7485

Which seems very impressive.

5. FURTHER ANALYSIS OF THE BEST MODEL

- The Support Vector Regression (Linear Kernel) model based on BoW features has 100 features out of which only 55 of them are non zero.
- The Linear Regression (L2 Regularized) model based on Word2Vec features has 200 features out of which only 86 of them are non zero.

Stats on test errors are,

- The minimum value is -9.91
- The maximum value is 9.39
- 75% of the errors are between -1.6875 and 4.6275

6. ADVANTAGES AND DISADVANTAGES OF THE MODELS

6.1. Linear Regression

Advantages:-

- It's a very simple model.
- It's a high bias model so there is less chance of overfitting.
- We can easily get feature importance with the absolute value of the weights.
- It's very fast because there aren't many operations involved.

Disadvantages:-

- It's very simple, so it can't capture complex patterns.
- Because it is a high bias model, it can underfit easily.
- It is affected by outliers. So we should use RANSAC.
- Feature importance can get affected because of collinearity.

6.2. Support Vector Machine

Advantages:-

- It only cares about the support vectors. So it is less affected by outliers.
- It works great for high-dimensional data.
- Because of the kernel, we can use it even when a similarity matrix is given as the data.

Disadvantages:-

- Choosing the right kernel is a hard task.

- It slows down heavily when the dataset is large.

6.3. K-NN Regressor

Advantages:-

- It is easy to understand.
- Even though it directly does not provide feature importance. It is interpretable when we view the neighbourhood data points.
- No training is required because it is instance-based.

Disadvantages:-

- It has high time complexity.
- It is affected by the curse of dimensionality.
- Since it is instance based we need to store the train data in memory.

7. REFERENCES

- AppliedRoots. [https://www.appliedroots.com/]
- Scikit-learn. [https://scikit-learn.org/stable/user_guide.html]
- XGBoost doc. [https://xgboost.readthedocs.io/en/stable/]