EE559 Final Project: Predicting the Online News Popularity

Data Set: Online News Popularity

Xin Zhang, xzhang@usc.edu

Xuan Shi, xuanshi@usc.edu

Date: 05/08/2021

1. Abstract

As we march into an ever-advanced era, people spend the majority of time surfing the internet. Social media communication takes the place of traditional interacting between people such as a letter. This change allows people to receive and send information instantly. In this situation, predicting the popularity of online news becomes ever more necessary as before. The News department wishes to know what kind of news could attract more people to read and to share. And it is important to predict the number of shares before they publish some news, so that they could improve their news before publication.

In this project, we were given a heterogeneous set of features about articles collected from Mashable websites, and our goal of this task is to predict the number of shares (popularity) of each article.

There are many techniques in this project, but could generally be categorized as preprocessing, dimension reduction or feature reduction, Regression models, and performance measurements. We implemented the preprocessing technique to normalize and standardize the data. We used the Selection from Model that measures the most important weight and PCA to reduce the dimension of data. And for the regression model, we designed Linear Regression, Ridge Regression, Lasso Regression, Support Vector Regression in Linear and RBF kernel, and RBF network Regression model to predict. To test our result, we would use some performance measurements such as Mean Absolute Error (MAE), Coefficient of determination (R^2), pMSE, pMAE, and Modified R^2. In the last, by comparing those models with cross validation enabled us to choose the best model.

In conclusion, the best regression model on this project is Support Vector Machine with linear kernel and optimal C* = 2.72, and preprocessing data by standardization with our technique that would be introduced later. Our code could be accessed here[1].

[1] GitHub Repository: https://github.com/xzhang603/EE-559-Online-News-Popularity-

2. Introduction (Problem Statement and Goals)

The training data set consists of 35645 news articles from Mashable websites. Moreover, there are no missing values. And the source was published on UCI Machine Learning Repository as https://archive.ics.uci.edu/ml/datasets/online+news+popularity. For each new article, it consists of 58 features, such as Number of words in the article and Article category (Mashable data channel). And the label (number of sharing) of the training data set spread from 1 to over 800,000. The testing data set consisted of 2000 news articles and the dimension was the same as the training data.

Because this is a very huge data set that slows our training down if we choose a model with a large degree of freedom. In this case, our goal is to, by comparing different machine learning and preprocessing techniques, get a minimum error model and a fastest model with relatively small error by performance measurements.

3. Approach and Implementation

3.1. Dataset Usage

As we mentioned in the previous section, the training data set consists of 35645 data points with 58 features and corresponding labels. The testing data set consists of 2000 data points with 58 features and corresponding labels. In most of the procedures of our project, we used test data to compare the results and cross validation to select parameters.

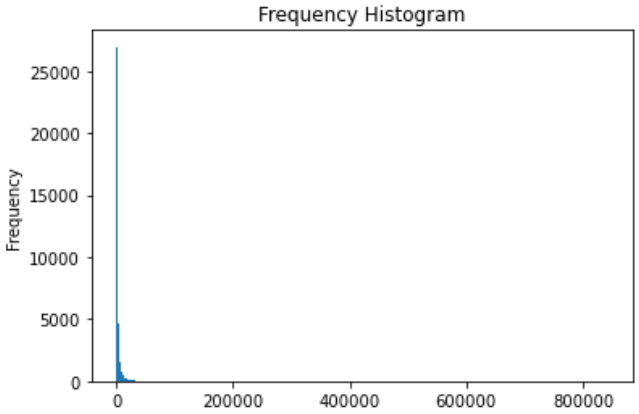

Figure 3.1.1: Frequency Histogram of Shares in Training Set

At the beginning when we stepped into the project, we found that the majority shares of the data were spread between 1,000 to 10,000. Only a very small part of data scatters in the range of 10,000 to 800,000, as shown in the Figure 3.1.1 above. The distribution of the data is typically long-tailed. So, we decided to fix the data outliers that only focus on the data in 2 standard deviations with mean of the data. After eliminating the data by fixing outliers, the data spread as Figure 3.1.2 below.

And this new data set covered 98% of original data points, which consists of 35103 data points total.

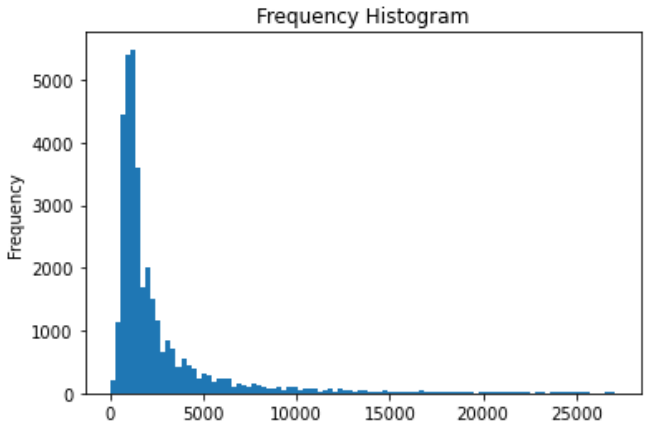

Figure 3.1.2: Frequency Histogram of Shares in Training Set after Fixed Outliers

However, after we finished our first version of the model, we found that our model could not predict that article news that has a large number of shares. If we predict an 800,000 shares article as 5,000 shares, each performance measure would increase dramatically when the number testing data is small. So, we will compare whether to reserve this technique in Section 4.

We use Cross Validation in parameter selections of RBF network models arranged as nested loops. At the beginning, we set the parameter candidates as a set. Then, running the cross validation to get the optimal K* and γ^*. Cross validation is a model evaluation method. K-fold cross validation is one way to improve over the holdout method. The data is divided into k subsets, and the hold out method is repeated k times. Each time, one of the k subsets is used as the validation set and other k-1 subsets are put together to form a training set. Then the average error across all k trails is computed [1]. By dividing the training into **5** folds. We calculate the MAE of each time and get an average MAE that is assigned to the current parameter. And get the minimum average MAE from all parameters. We use it to find the gamma and number of k clusters of the RBF network.

The training data set (features and label) was used all over the project from preprocessing, feature dimension reduction, and training. The testing data were used after getting the training model such as preprocessing parameters from the training set. The validation data was used when we did 5-folds cross validation.

We use test data multiple times to compare different model results in the end. Also, the test data was used every iteration when selecting the parameter selections of Ridge regression, SVC kernel parameters.

3.2. Preprocessing

Feature	Type (#)
Words	
Number of words in the title	number (1)
Number of words in the article	number (1)
Average word length	number (1)
Rate of non-stop words	ratio (1)
Rate of unique words	ratio (1)
Rate of unique non-stop words	ratio (1)
Links	
Number of links	number (1)
Number of Mashable article links	number (1)
Minimum, average and maximum number of shares of Mashable links	number (3)
Digital Media	
Number of images	number (1)
Number of videos	number (1)
Time	
Day of the week	nominal (1)
Published on a weekend?	bool (1)

Feature	Type (#)
Keywords	
Number of keywords	number (1)
Worst keyword (min./avg./max. shares)	number (3)
Average keyword (min./avg./max. shares)	number (3)
Best keyword (min./avg./max. shares)	number (3)
Article category (Mashable data channel)	nominal (1)
Natural Language Processing	
Closeness to top 5 LDA topics	ratio (5)
Title subjectivity	ratio (1)
Article text subjectivity score and its absolute difference to 0.5	ratio (2)
Title sentiment polarity	ratio (1)
Rate of positive and negative words	ratio (2)
Pos. words rate among non-neutral words	ratio (1)
Neg. words rate among non-neutral words	ratio (1)
Polarity of positive words (min./avg./max.)	ratio (3)
Polarity of negative words (min./avg./max.)	ratio (3)
Article text polarity score and its absolute difference to 0.5	ratio (2)

Target	Type (#)
Number of article Mashable shares	number (1)

Figure 3.2.1: Feature data Categories

Because of the diversities of data, applying preprocessing techniques before training is important. As we could see in Figure 3.2.1 above, there are three types of features: number, nominal, and ratio. The categorical features have been already translated into nominal (one-hot) data, and the boolean data could be thought of as nominal data. So, normalization and standardization could be helpful here. It is often beneficial if different features have different ranges of values. The standardization equation shows below in equation (1). However, those nominal features typically could not be standardized.

$$Z = \frac{x - \mu}{\sigma} \quad (1)$$

Also, there are some features that have an extremely large width (range) that others. In this case mapping the data into log range is a good practice.

$$\bar{x} = log(n - x) \quad (2)$$

In this equation, n = 1 if $min\{x_j\} = 0$, and then use x_j as a feature value.

In the comparison section, we would decide whether to choose to use standardization or not. We compared the result that implements and not

```
The features that have large variance is:
{'kw_min_max', 'kw_min_avg', 'self_reference_min_shares', 'self_reference_avg_sharess', 'self_r
eference_max_shares', 'kw_max_min', 'kw_max_max', 'kw_avg_max', 'kw_avg_avg', 'kw_max_avg'}

The features that is one-hot is:
{'data_channel_is_world', 'data_channel_is_entertainment', 'data_channel_is_tech', 'weekday_is_
monday', 'weekday_is_saturday', 'weekday_is_sunday', 'data_channel_is_socmed', 'is_weekend', 'w
eekday_is_thursday', 'n_non_stop_words', 'weekday_is_wednesday', 'weekday_is_friday', 'data_cha
nnel_is_lifestyle', 'data_channel_is_bus', 'weekday_is_tuesday'}
```

implements the preprocessing techniques. In Figure 3.2.2, it shows which features have large variance and which features are nominal.

Figure 3.2.1: Large Variance and One-Hot Features

3.3. Feature dimensionality adjustment (if applicable)

Since there are 58 features in the data set. The data set must include some features that are not that important or even useless. Applying feature dimensionality adjustment techniques could not only reduce the operation time complexity but also avoid overfitting. It makes sense to use feature selection and dimension reduction.

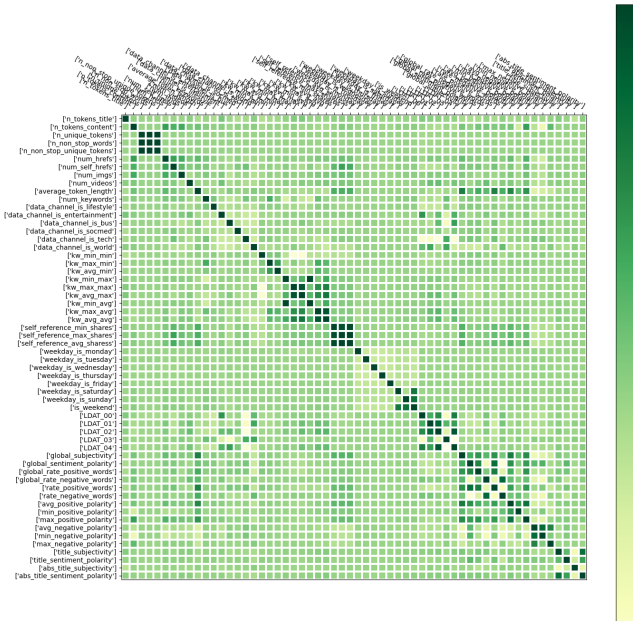

Figure 3.3.1: Correlation Matrix of Online News Popularity

To better understand the dependency between features, we compute and visualize the correlation among 58 provided features, which indicates in the Figure 3.3.1 above. According to the correlation matrix, we can figure out that provided features are almost independent of each other, except some groups, such as:

{‘n_unique_tokens’,‘n_non_stop_words’, ‘n_non_stop_tokens’}
{‘self_reference_min_shares’, ‘self_reference_max_shares’, ‘self_reference_avg_shares’},
{‘kw_avg_avg’, ‘kw_max_avg’}

Although it is not impossible to reduce feature dimensions according to the correlation matrix, we could try feature selection and dimension reduction techniques to eliminate the dependency and compare the performance of which techniques works better on the data set.

For the feature selection, we use SelectFromModel in sklearn. SelectFromModel is a meta-transformer that can be used alongside any estimator that assigns importance to each feature through a specific attribute (such as coef_, feature_importance_) or via an impotance_getter callable after fitting. The features are considered unimportant and removed if the corresponding importance of the feature values are below the provided threshold parameter of default parameter. Apart from specifying the threshold numerically, there are built-in heuristics for finding a threshold using a string argument [2]. We implemented SelectFromModel in Linear Regression, Ridge Regression, Lasso Regression, and Support Vector Regression models. Due to the fact that our designed RBF network Regression model does not have those specific attributes of feature importance, we did not use it on the RBF network model.

The dimension reduction method we used is Principal Component Analysis (PCA). Large datasets are increasingly common and are often difficult to interpret. Principal component Analysis is a technique for reducing the dimensionality of such datasets, increasing interpretability but at the same time minimizing information loss. It does so by creating new uncorrelated variables that successively maximize variance [3]. The PCA was implemented before doing regression, especially in RBF network regression. Moreover, we used PCA before other regression models but after SelectFromModel. The Figure 3.4.1 below further shows how PCA works and why it is useful.

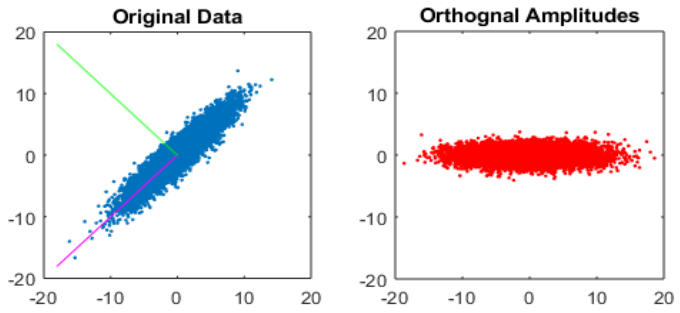

Figure 3.3.2: How Principal Component Analysis Works

3.4. Training, classification or regression, and model selection

Note: all the functions we used were from scikit-learn except our designed RBF Network.

3.4.1 Linear Regression Model

First, we adopt the linear regression, a classical regression model in machine learning, to predict the popularity of online news. Given the input data x with d-dimensions, the linear regression model predicts the result by multiplying the trainable weight w with augmented input X: $\widehat{y} = Xw$, where $\widehat{y}$ is the prediction of the input.

To compute the weight w, we usually employ least squares method to optimize the cost function (3), where N is the number of samples, and y is the ground truth label.

$$J(w) = \frac{1}{N} || Xw - y ||_2^2 \quad (3)$$

The d.o.f of the Linear Regression model is $D + 1$, depending on the dimension of the input feature.

3.4.2 Ridge Model

To restrain the range of the value of weights in a Linear Regression model, we further introduce the Ridge Model in our experiment, adding a restriction item of weights on the cost function. The cost function of the Ridge Model is in equation (4) , in which the α is a hyper-parameter to adjust the weight of the restriction item in the total loss.

$$J(w) = \frac{1}{N} ||Xw - y||_2^2 + \alpha ||w||_2^2 \quad (4)$$

To select the optimal α, we conduct the model selection with candidate parameters ranging from e^{-20} to e^{4}, interpolated in a log scale. And test them on test data to get the optimal α.

3.4.3 Lasso Model

Lasso, or Least Absolute Shrinkage and Selection Operator, is quite similar conceptually to Ridge Regression. It also adds a penalty for non-zero coefficients, but unlike ridge regression which penalizes sum of squared coefficients (*L2* penalty), Lasso penalizes the sum of their absolute values (*L1* penalty) [5]. The equation of Lasso shows below. We used the default parameter of this model.

$$J(w) = \frac{1}{2*N} ||Xw - y||_2^2 + \alpha ||w||_1 \quad (5)$$

3.4.4 SVR

Support Vector Regression is a model that was created based on the Support Vector Machine. It is reasonable to think that SVR is SVM adapted to a regression problem. SVR gives us the flexibility to define how much error is acceptable in our model and will find an appropriate line (or hyperplane in higher dimensions) to fit the data. The criterion function of SVR is to minimize the coefficients, the *L2-norm* of the coefficient vector. The error term is instead handled in the constraints, where we set the absolute error less than or equal to a specified margin, called the maximum error ϵ[4]. The equation (6) detailed shows the interpretation of error function.

$$J(w) = C \sum_{n=1}^{N} E_\epsilon [\widehat{y_n(x_n)} - y_n] + \frac{1}{2} ||w||_2^2 \quad (6)$$

Support Vector Regression are defined as the vectors that are outside the ϵtube or on the boundary of theϵtube. It tells us about how tolerance could accept the error and predict the result.

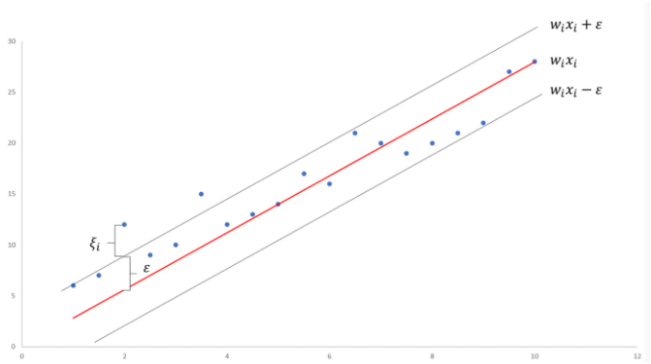

Figure 3.4.1: Example of SVR

In our kernel selection, we choose linear and RBF kernels to train our data. The kernel could be thought of as a window function that inside the range of the window would be calculated, so that outside the window would be ignored. The equations of linear and RBF kernels are shown below in equation (7) and (8).

$$k(x_i, x_j) = x_i^T x_j \quad (7)$$

$$k(x_i, x_j) = exp(-\gamma ||x_i - x_j||^2) \quad (8)$$

To select the optimal value of C^* in linear kernel SVR, we conduct the model selection with candidate parameters ranging from 10^{-3} to 10^{3}, interpolated in a log scale. And test them on test data to get the optimal C.

3.4.5 RBF

Due to the intractability of data distribution, we adopt a sophisticated model, Radial-Basis Function (RBF), which is expected to achieve better performance.

Generally, the RBF is composed of two layers: the first layer is to evaluate the distance $\phi_m = exp(-\gamma||x - \mu_m||_2^2)$ between input features x and centers μ_m, while the second layer is to transform the output of the first layer to the final prediction through a feedforward artificial neural network (ANN). In the computation of the ϕ_j, γ is an essential hyper-parameter. In this report, we choose K-Means algorithm to generate centers $\{\mu_1, \mu_2, ..., \mu_M\}$, where M is a hyper-parameter to denote the number of centers. Thus, in the model selection for RBF, there are two hyper-parameters that need to be specified for the optimal model.

In the optimization of the RBF, we adopt the cross validation to select the optimal parameter of K^*(centers) and γ^*. The first layer is trained in an unsupervised method, while the second layer is trained with MSE in a supervised method. Both the input of the first and second layer are augmented.

The d.o.f RBF is $M(D+2)+1$, the summation of the d.o.f of the first layer $M(D+1)$ and the second layer $M+1$. Because we want the constraints $N_c \simeq (3-10)\,d.o.f$, we choose the range of the number of centers from {30, 50, 100, 150, 200} as candidate parameters. And when we get a specific small range of centers that works great, we would further limit the centers in the range from optimal $K-10$ to optimal $K+10$ with steps of 2. For the value of γ, we calculate an approximate default spacing in equation (9). And we search the value of $\gamma$with the factor of 2 below and above the default value. After some attempts, we found that the $\{\frac{\gamma}{1024}, \frac{\gamma}{512}, \frac{\gamma}{256}, \frac{\gamma}{128}\}$is a relatively proper range for our training.

$$\gamma = \left(\frac{\Delta_{x1}\Delta_{x2}\cdots\Delta_{xD}}{M}\right)^{\frac{1}{D}}, \text{ where D is the dimension of features} \quad (9)$$

3.5. Performance Measurements and Comparison Techniques

(1) Mean Absolute Error (MAE): determines the acuteness of the predictions. It is scale dependent.

MAE = $\frac{1}{N}\sum_{n=1}^{N}|y_n - \widehat{y_n}|$, where y_n is the ground truth, $\widehat{y_n}$ is the predicted value.

(2) Coefficient of determination (R-Squared, R^2): usually ranges from 0 to 1, representing the goodness-of-fit. A high R^2 value shows the predictions match the observed values well. A negative value means the system performs worse than the trivial system.

$R^2 = 1 - \frac{\sum_{n=1}^{N}(y_n - \widehat{y_n})^2}{\sum_{n=1}^{N}(y_n - \bar{y})^2}$, where $\bar{y}$ is the mean value of all y_n.

(3) PMSE: removes the effect of large scale target values.

$pMSE = \frac{1}{N}\sum_{n=1}^{N}(\frac{y_n - \widehat{y_n}}{r + y_n})^2$, where r = 10

(4) PMAE: similar to PMSE, but use MAE as the base

$pMAE = \frac{1}{N}\sum_{n=1}^{N}|\frac{y_n - \widehat{y_n}}{r + y_n}|$, where r = 10

(5) Modified R^2:

$mR^2 = 1 - \frac{pMSE}{\frac{1}{N}\sum_{n=1}^{N}(\frac{y_n - \widehat{y_n}}{r + y_n})^2}$, where r = 10

4. Analysis: Comparison of Results, Interpretation

4.1. Trivial system

	MAE	R2	pMSE	pMAE	mR2
vanilla/standard	2946.3742	-0.0016530	10.46015	1.8565	-0.2012
filter	2520.6835	-0.0069067	5.654991	1.3056	0.3506

To have a better understanding of the entire comparison system, we introduce the trivial system to compare our results. We will not consider those model performances worse than the trivial system filtered outliers result. Because the trivial system's prediction result is the mean of the label in test data. The reason that the results of

vanilla/standard and filter data are different is that filter data removes some data points that lie out of the outliers. The test result is from test data.

Vanilla data: the original data provided
Standard data: the data after preprocessing (standardize, use logspace to reduce large variance)
Filter data: the data that was removed the outliers
Select_feature data: those data could be reduced features by SelectFromModel

4.2. Linear Regression (Baseline System)

	MAE	R2	pMSE	pMAE	mR2
vanilla	2801.0519	0.0444990	6.814368	1.573098	0.217480
standard	12493466.2126	-3865377.4837	198324889.0465	9788.9495	-22774403.9015
filter	2373.0100	0.041355	3.599846	1.112598	0.586616
select_feature	2882.1048	0.008312	7.794960	1.730556	0.104875
filter + select_feature	2473.2342	0.005534	4.201341	1.234673	0.517544
all	7984085.9288	-1578614.1337	80948528.6634	6255.0969	-9295627.8887

The result implies that except the standard data, the rest of data sets have better results than trivial systems. The best data structure of Linear Regression is the filter outliers data, which MAE, R2, pMSE, pMAE, and mR2 shows in orange highlight color. The test result is from test data, and the model was trained from training data.

4.3. Ridge Regression

	MAE	R2	pMSE	pMAE	mR2
vanilla	2800.5746	0.044507	6.808057	1.572674	0.218204
standard	5063.9136	-0.670731	32.799790	3.300290	-2.766525
filter	2373.2167	0.041281	3.586720	1.113007	0.588123
select_feature	2882.1048	0.008312	7.794960	1.730556	0.104875
standard+filter+select_feature	2190.2399	-0.017259	2.020927	0.761998	0.767929
standard+select_feature	3597.5210	-0.14459	15.59546	2.24240	-0.79088
filter+select_feature	2421.6175	0.018147	4.412289	1.176593	0.493320
standard+filter	2393.6951	0.020903	3.788599	1.134882	0.564941

standard+filter+pca (optimal num_components= 15)	2515.7094	-0.010110	5.094067	1.285129	0.415029

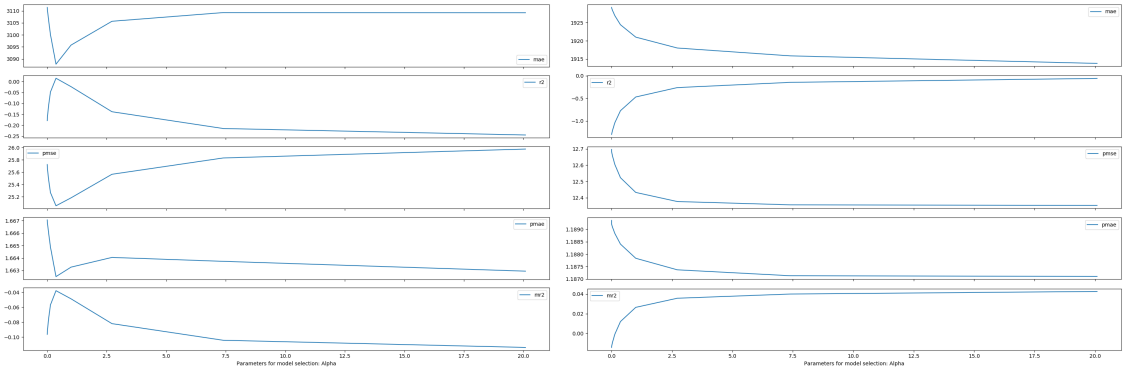

Figure 4.4.1: Alpha Selection of Ridge Regression by Cross Validation

The result of Ridge Regression has better results than linear regression, which MAE was decreased by 200. The best data structure of Ridge Regression is standardization + filter outliers + Feature reduction, which MAE, R2, pMSE, pMAE, and mR2 shows in orange highlight color. The optimal alpha* is equal to 20. The test result is from test data. The Figure 4.4.1 shows the alpha selection by cross validation. The left figure is based on Vanilla data, and the right figure is on Standardized data. The test result is from test data, and the model was trained from training data.

4.4. Lasso Regression

	MAE	R2	pMSE	pMAE	mR2
vanilla	2832.3017	0.0412859	7.458600	1.651043	0.143500
standard	2841.8166	0.022784	7.368488	1.650794	0.153848
filter	2424.5759	0.031560	4.149979	1.188481	0.523442
select_feature	2832.3017	0.0412858	7.458600	1.651043	0.143500
std + filter	2408.6377	0.0226423	3.989008	1.160601	0.541927
std + filter + select_feature	2408.6377	0.022642	3.989008	1.160601	0.541927
fil + sele	2424.5759	0.031560	4.149979	1.188481	0.523442
std + sele	2844.9117	0.021102	7.434582	1.650150	0.146259

The result of Lasso Regression performs worse than Ridge Regression. The best data structure of Ridge Regression is standardization + filter outliers + feature reduction, which MAE, R2, pMSE, pMAE, and mR2 shows in orange highlight color. The input value of Lasso Regression is the default value. The test result is from test data, and the model was trained from training data.

4.5. Support Vector Regression

	MAE	R2	pMSE	pMAE	mR2
vanilla(default C)	31907.7174	-149.7793	4968.2441	22.549129	-569.5225
standard(default C)	2124.9422	-0.051033	1.144368	0.572920	0.868588
filter(default C)	65568.6983	-160.5631	8589.9888	53.298871	-985.421242
select_feature(default)	2155.8044	-0.060824	1.146082	0.585044	0.868391
standard(search) [Kernel: Linear, C:2.72]	2122.5175	-0.049489	1.159962	0.573204	0.866797
standard + fil(search) [Kernel: Linear, C:2.72]	2124.9592	-0.052807	1.145537	0.565710	0.868453
standard + fil [Kernel: rbf]	2170.0206	-0.0700500	1.134293	0.583859	0.869744
standard [Kernel: rbf]	2168.8783	-0.068269	1.161575	0.590457	0.866611

The result of Linear Kernel SVR performs better than ridge regression. The best data structure of Linear Kernel SVR is standardization, which MAE, R2, pMSE, pMAE, and mR2 shows in orange highlight color. The Optimal C* is 2.72. The test result is from test data, and the model was trained from training data. Because the result of standardization + filtered outliers and standardization are close, we upload both of them on Vocareum.

The result of RBF Kernel SVR shows in the last two rows of the chart above. The best data structure of RBF Kernel SVR is standardization alone, which MAE, R2, pMSE, pMAE, and mR2 shows in red highlight color. Although it is better than other regression models, the RBF Kernel's result is slightly worse than Linear Kernel. The parameters of RBF Kernel SVR are defaulted. The test result is from test data, and the model was trained from training data.

4.6. RBF Network Regression

	MAE	R2	pMSE	pMAE	mR2
vanilla	2946.3743	-0.001653	10.460154	1.856517	-0.201179
standard (M=50, g=0.01)	2867.6008	0.025094	7.279845	1.681571	0.164028
filter (M=30, g=0.03)	2520.6835	-0.006907	5.654991	1.305600	0.350616
standard+ filter (M:200, g:0.001918)	2396.8486	0.029565	3.713570	1.135885	0.573557

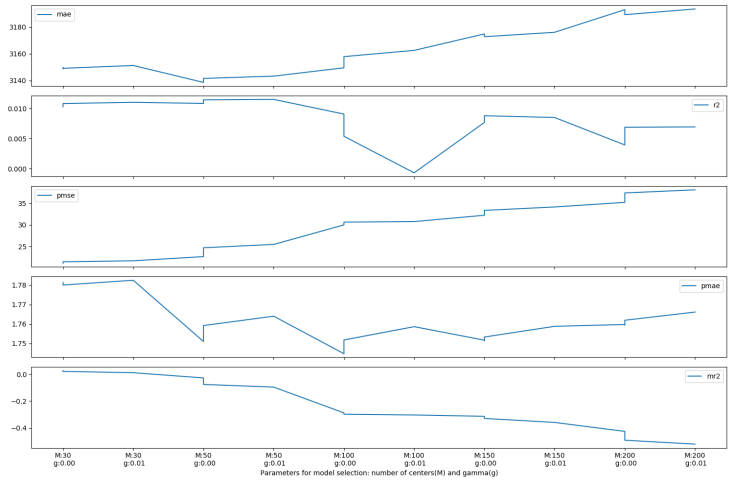

Figure 4.6.1: Center and γSelection on RBF Network Regression

The result of RBF Network Regression performs not as well as expected. The best data structure of RBF Network Regression is standardization + filter outliers, which MAE, R2, pMSE, pMAE, and mR2 shows in orange highlight color. The Optimal center K* = 200, and the γ^* = 0.001918. The test result is from test data, and the model was trained from training data. In Figure 4.6.1, the result is from 5-folds cross validation data.

4.7. Vocareum Result

Based on the Result of above 6 predictor, we finally choose Support Vector Regression with linear Kernel, and optimal C* = 2.72. The final performance that we uploaded the final predict.py file on Vocareum is:

public_pMAE_score: 0.672
public_pMSE_score: 1.683
The final result is slightly worse than the result of predicted testing data. It might be because the majority of training data is spread in 1,000-10,000 numbers of shares. If a lot of data on Vocareum has more numbers of shares like 100,000+, our model would not predict the shares as expected. If we have more 100,000+ shares of training data, the result would be better. Also, if we classify data to popular and unpopular before regression and then do regression on each class, the final performance would also be better. More improvement ideas will be introduced in Section 6.

5. Contributions of each team member

Xuan Shi:

- Built PCA and correlation matrix
- Designed the RBF model and Final Prediction model that would upload on Vocareum.
- Integrated the entire project into a main function
- Wrote final report

Xin Zhang:

- Designed the process of the entire project
- Designed the preprocessing technique and feature selection
- Implemented the Linear, Lasso, Ridge and SVR model
- Wrote final report

6. Summary and conclusions

After training and predicting the entire data set, we concluded that for the data set of Online News Popularity, the outstanding machine learning method of regression is Support Vector Regression with linear kernel and optimal C* = 2.72. The data should be standardized with our technique and choose whether to use filter outliers. However, if more data would be added into this data set, we also recommend using Ridge Regression with optimal alpha* = 20. Although the result of SVR would be better than Ridge Regression, the time complexity or degree of freedom of Ridge Regression is much less than SVR, which illustrates that it would calculate faster than SVR. It would be a tradeoff of accuracy and time complexity.

The performances of each method are not ideal because of the large range of shares. To figure this problem out could first classify the data into two or three categories which range from highest popularity to lowest. Then, implement the regression models on each classification.

The project we did was a team project. We learned a lot not only from the knowledge of machine learning models but also from the procedure of team work. It allowed us to learn to break a complex problem into parts and steps, and share the diversity of perspectives. Lastly, we were grateful to have this opportunity to do such a remarkable project and have an outstanding teammate.

References

[1] Schneider, Jeff. *Cross Validation*, CMU, 7 Feb. 1997, www.cs.cmu.edu/~schneide/tut5/node42.html.

[2] “Sklearn.feature_selection.SelectFromModel¶.” *Scikit*, scikit-learn.org/stable/modules/generated/sklearn.feature_selection.SelectFromModel.html#sklearn.feature_selection.SelectFromModel.

[3] Jolliffe Ian T. and Cadima Jorge, et al. “Principal Component Analysis: a Review and Recent Developments.” *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 13 Apr. 2016, royalsocietypublishing.org/doi/10.1098/rsta.2015.0202.

[4] Sharp, Tom. “An Introduction to Support Vector Regression (SVR).” *Medium*, Towards Data Science, 6 May 2020, towardsdatascience.com/an-introduction-to-support-vector-regression-svr-a3ebc1672c2.

[5] Oleszak, Michał. “(Tutorial) Regularization: Ridge, Lasso and Elastic Net.” *DataCamp Community*, 12 Nov. 2019, www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net.