Chance of Graduate Admission prediction

*Prediction of USA graduate admission for Indian undergraduate appliers

Run Zeng
Department of Electrical and Computer Engineering
University Of Waterloo
Waterloo, Canada
r24zeng@uwaterloo.ca

***Abstract*—This paper mainly introduces the whole procedure of my project on possibility prediction of US graduate admission from Indian students' prospective. The dataset I use is on Kaggle (*https://www.kaggle.com/mohansacharya/graduate-admissions*). I mainly used Linear Regression, Ridge Regression, Multi-layer Perceptron Regression (MLP) and Random Forest Regression. The difference I made to this problem is the model ensemble by stacking. Mean square error (MSE) and r2-score are the main metrics to analyze modeling performance. At the same time of introducing my project, I briefly introduced the principles of regression methods used in my project and common used evaluation metrics. In this paper, I also provide many figures of my results.**

***Index Terms*—Linear Regression, Ridge Regression, MLP, Random Forest Regression, MSE, r2, data preprocessing, model ensemble**

I. Introduction

Machine learning mainly solves classification and regression problems, which is an important research area and very useful in making decision. Both of classification and regression algorithms aim to evaluate unknow data by training labeled data sample. But they are still different. The output of the former is discrete, which means we separate data to a few classes by classification algorithm then evaluate new coming data belongs to which class. The output of the latter is continuous, which means we learn from old data then evaluate the probability of an event happening.

In this paper, I will introduce my project " Prediction of USA graduate admission for Indian undergraduate appliers" based on regression algorithms. The main content is to show the process from analyzing data, making model to result evaluation. At the same time, I will introduce the principles of the methods used in my project. In section 2, I will explain my dataset and introduce the research background. In section 3, I will give brief introduction about how I organize my project and the main algorithm (Linear Regression) I used in my project. In section 4, the completed work line will be provided including selecting base model and other models fitting, and the methods of data preprocessing, cross validation and best parameter search will also be mentioned. In section 5, I will provide details about how to optimize model by stacking and analyze the results. In section 6, I will conclude the contribution I made to this project, the challenges I faced in the process, the existed problems and possible solutions.

II. Literature Review

The dataset comes from Kaggle website (*https://www.kaggle.com/mohansacharya/graduate-admissions*), which contains 500 students data and includes 7 features as predictors such as "GRE score", "TOEFL score", "University Rating" (Indian undergraduate school), "SOP" (Statement of Purpose), "LOP" (Letter of recommendation strength), "CGPA" (out of 10), "Research experience", and 1 column as target"Chance of Admission" (float datatype, ranging from 0 to 1). "Chance of admit" were evaluated by appliers.

This dataset has various use. Many people have provided solutions to this problem. Most people take this problem as classification problem (The student can be admitted or not), generally, they get very high f1-score (up to 96%) [1]. Some people take it as admission line case. They analyze what is the minimal line for every feature to make the admission possibility over 90%. For example [2], if a student wants to get 93.5% admission chance, the scores or rates must satisfy GRE $>$ 332.8, TOEFL $>$ 116.2, University Rating $>$ 4.6, SOP $>$ 4.5, LOR $>$ 4.5, CGPA $>$ 9.5, Research $=$ 1.

Only few people take it as regression problem to predict the possibility of admission [3]. These paper show that they built some basic models to get reasonable results and use cross validation to select the best feature combination. The most common used algorithms are Linear Regression, Decision Tree Regression and KNN Regression. Linear Regression performs the best among them. Somebody tried to build deep Neural Networks to achieve 92.8% accuracy in validation data instead of using direct frameworks and libraries. But few people ensemble models to improve the prediction accuracy [4].

All the projects are processing in Python. Importing numpy and pandas library to deal with data. Matplotlib and seaborn associate analysis by visualizing data. Sklearn is the strongest library to call popular algorithms package to train data and evaluate results. Although these tools have some limitations

and not super flexible, but for basic usage they are enough. After comparing a few algorithms, I decided to start with Linear Regression building my base regressor. It is also a very popular method in regression problem. Therefore, I will introduce the principle of regression by going through more details about Linear Regression.

III. Theory of Linear Regression

In machine learning, any problem can be taken as classification problem or regression problem. Different from classification, the prediction value is always continuous in a certain range, simply to say, target value is the probability of one event happening. For example, a medicine institution wants to diagnose a patient according to known a set of data of patients health records. If we want to know the patient is ill or not, this is a classification problem. But sometimes the result is not absolute. Before getting ill, the patient wants to know the probability of being ill, which can be considered as regression problem.

Linear regression is the simplest statistic model to imply the relation between variables. Firstly, let us only consider two variables relation, then I will extend it to multi variables linear regression [5].

A. Simple Linear Regression

Assume we have two variables, x and y. x is called as independent variable or predictor variable, y is called as dependent variable or response variable. In order to find the linear relation between them, I set a line to fit function them as follows:

$$y = \beta_0 + \beta_1 x$$

β_0 is the intercept of the line. β_1 is the slope of line, positive value indicates positive relationship, negative value indicates negative relationship. The best line is to fit all data. But sample data can not be without noise so as to our model is impossible to fit all data perfectly.

To adjust the noise, we assume $y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$, (x_i, y_i) is the pair of true data, ξ_i is the noise which represent why the line can not fit perfectly. We will set ε_i as Gaussian noise: $\varepsilon \backsim \mathcal{N}(0, \sigma^2)$. β_0, β_1, σ are fix values in this equation.

How to know this line fits or not or how to define a good line? Obviously, our goal is to let the difference between true y and test y be as small as possible. This concept can be defined as:

$$\min_{\beta_0, \beta_1} : \sum_{i=1}^{n} [y_i - (\beta_0 + \beta_1 x_i)]^2$$

We are looking for β_0 and β_1 making the sum of square error minimal. In order to solve this, we introduce least-squares linear regression.

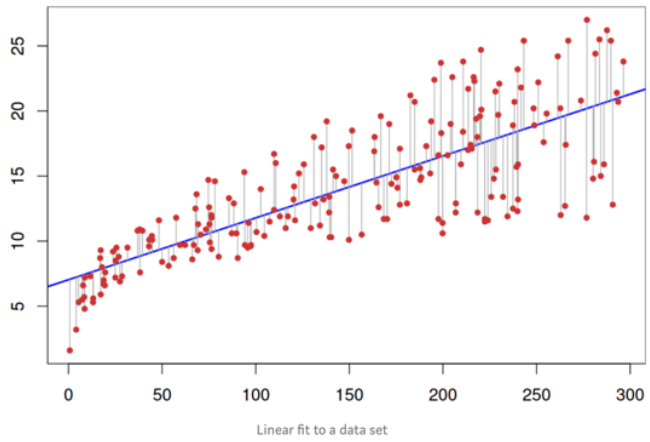


Fig. 1. Linear regression model

After a series of math computation, the solution is :

$$\begin{aligned}\hat{\beta}_1 &= \frac{\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2} \\ &= r\frac{s_y}{s_x}, \\ \hat{\beta}_0 &= \bar{y} - \hat{\beta}_1 \bar{x},\end{aligned}$$

In above equation, r is called as correlation coefficient, which is a very crucial value, representing how strong y is relevant to x.

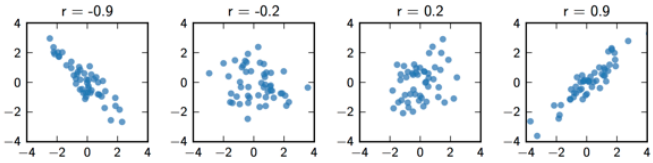


Fig. 2. r2 shows the correlation between independent and dependent variables

When r is closed to 1 or -1, variable x and y have strong relationship, positive or negative. If r is closed to 0, the relation is very weak.

B. Multiple Linear Regression

After talking about one independent variable as input, multiple variables are the more common case.

We set vector $(x_i, \ldots, x_p)$ for data point i, which means every data i has p features to contribute target y. Then we also use a line to fit them:

$$\mathbf{Y} = \beta_0 + \beta_1 \mathbf{X}_1 + \beta_2 \mathbf{X}_2 + \cdots + \beta_p \mathbf{X}_p$$

We represent input data as an $\mathbf{X}_p$ of matrix $\mathbf{X}$. Adding a noise as before, this equation can be adjusted as:

$$\mathbf{Y} = \beta \mathbf{X} + \varepsilon$$

Then this problem can be transformed as an optimization problem:

$$\min_{\beta} \sum_{i=1}^{n} (y_i - \mathbf{X}_i \beta)^2$$

Find the β to minimize the sum of square error, x_i means the columns of matrix $\mathbf{X}$. Then we use linear algebra to solve this.

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^{\mathbf{T}}y$$

C. Assess the accuracy of model

When we find a line through math, how do we evaluate the performance of this model? Image our prediction $\hat{y}_i$ of every point ($\mathbf{X}_i$, y_i) is very close to mean y, then our line will be nearly parallel to the x-axis. Once this happens, y_i does depend on $\mathbf{X}$. For sure, we do not want this to happen. Then we hope our prediction $\hat{y}_i$ is far from mean y but close to y_i. This concept can be defined as based on the figure below:

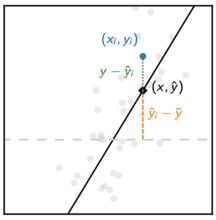


Fig. 3. Linear regression model

$$y_i - \bar{y} = \underbrace{(\hat{y}_i - \bar{y})}_{\textit{difference explained by model}} + \underbrace{(y_i - \hat{y}_i)}_{\textit{difference not explained by model}}$$

where (x_i, y_i) is the true value of sample data, $\hat{y}_i$ is the prediction, $\bar{y}$ is the mean of y. $y_i - \bar{y}$ represent the distance of y_i to $\bar{y}$. $y_i - \hat{y}_i$ is the residual error, which can not be explained by model. $\hat{y}_i - \bar{y}$ can be explained by model. Undoubtedly, we hope our model explain more of the difference, which means maximize left part but minimize the right part of the equation.

$$\underbrace{\sum_i (y_i - \bar{y})^2}_{SS_total} = \underbrace{\sum_i (\hat{y}_i - \bar{y})^2}_{SS_model} + \underbrace{\sum_i (y_i - \hat{y}_i)^2}_{SS_error}$$

$$1 = \underbrace{\frac{SSM}{SST}}_{r^2} + \underbrace{\frac{SSE}{SST}}_{1-r^2}$$

SSM stands for sum of squares of model difference, SSE stands for sum of squares of error, SST stands for sum of squares of total difference. According to what we analyzed above, the bigger of SSM but the smaller of SSE the better. $r2$ is called the regression coefficient of determination. In other words, $r2$ represent how much better your regression line is than a simple horizontal line through the mean of the data [6].

IV. Implementation

My project goes through four main steps: a) Data basic information analysis; b) Data preprocessing; c) Base model selection; d) Model training and testing. I will talk the details in the following paragraphs. In order to improve the accuracy, I add one last step: Model ensemble, I will talk about it in next section.

A. Data basic information analysis

I use pandas to load training data from csv file. Then check the features and shape of my data.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
Serial No.            500 non-null int64
GRE Score             500 non-null int64
TOEFL Score           500 non-null int64
University Rating     500 non-null int64
SOP                   500 non-null float64
LOR                   500 non-null float64
CGPA                  500 non-null float64
Research              500 non-null int64
Chance of Admit       500 non-null float64
dtypes: float64(4), int64(5)
memory usage: 35.2 KB
```

Fig. 4. Show info of data

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

Fig. 5. Show the head five rows of data

	SerialNo	GREScore	TOEFLScore	UniversityRating	SOP	LOR	CGPA	Research	ChanceOfAdmit
count	500.000000	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	250.500000	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	144.481833	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	1.000000	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	125.750000	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	250.500000	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	375.250000	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	500.000000	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

Fig. 6. Describe data

According to the figures above, we know that, data set contains 8 features, including "Serial No.", "GRE Score", "TOEFL Score", "University Rating", "SOP", "LOR", "CGPA", "Research", the column of "Chance of Admit" is target variable. There are 500 students' records in total. No data is missing.

"Serial No.", "GRE Score", "TOEFL Score", "University Rating", "Research" are integers, ranging from 1-500, 90 to 340, 92 to 120, 1 to 5 respectively. "SOP", "SOR", "CGPA",

”Chance of Admit” are floats, ranging from 1 to 5, 1 to 5, 6.8 to 9.92, 0.34 to 0.97 respectively.

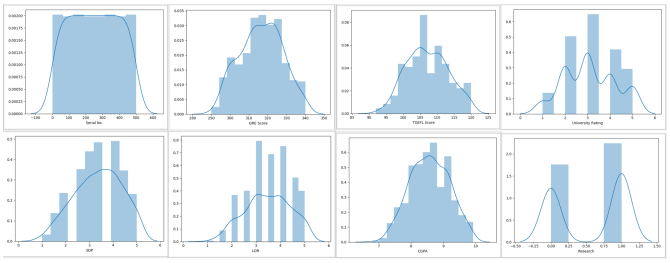


Fig. 7. Distribution of density of each feature

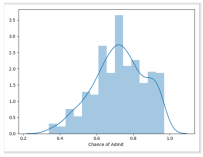


Fig. 8. Distribution of density of target

According to figure 7 and 8, except ”Serial No.” and ”Research”, all of them obey normal distribution. The shape of data indicates our data is balanced.

B. Data preprocessing

In this step we will transform our data to the form that benefits to training model. In order to index every feature easily, we rename all the features(remove all spaces between words). Let us to see the correlation of features by heatmap.

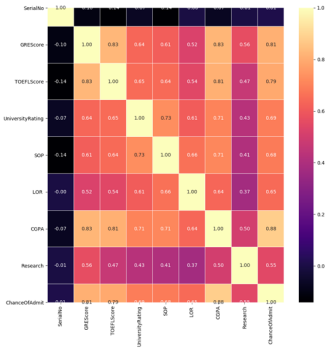


Fig. 9. Distribution of density of target

According to figure 7 and 9, ”Serial No.” should not be a feature, so we drop this column. From the heatmap we can see, ”Research” affects target ”ChanceOfAdmit” little, ”GREScore”, ”TOEFLScore”, ”CGPA” affect target most. Among features, ”GREScore”, ”TOEFLScore”, ”CGPA” are very relevant to each other. Because there are not too many features, we do not need to throw any feature.

One of the most important step is to split data sample into training data and validation data. The reason is, model fits training data perfectly but fails on testing data sometimes. This situation is caused by overfitting. This can be explained as figure 10. If this model also works well on validation data, it is a good model. We randomly abstract 20% as validation data, 80% as training data, separate the first 7 columns as independent variables X, the last columns as target y.

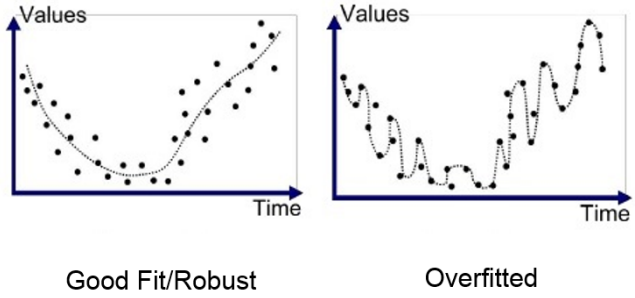


Fig. 10. Overfitted and Good fitted example

Normalize data is also an important step as data ranging differs very large will lead to big bias in model training. So I normalize all data to range from 0 to 1.

After these data preprocesing, we will have four set of data, training data X, training data y, validation data X, validation data y. Target y is not changed. Training data X looks like follows:

	GREScore	TOEFLScore	UniversityRating	SOP	LOR	CGPA	Research
249	0.62	0.678571	0.50	0.625	0.750	0.650641	1.0
433	0.52	0.678571	0.75	0.750	1.000	0.557692	0.0
19	0.26	0.357143	0.50	0.625	0.500	0.544872	0.0
322	0.48	0.535714	0.25	0.375	0.750	0.471154	0.0
332	0.36	0.500000	0.50	0.625	0.375	0.451923	1.0

Fig. 11. Head five rows of training data X

C. Base model

The most common used basic algorithms for regression are Logistic Regression, Linear Regression and Polynomial Regression. They have themselves' advantages and disadvantages. I choose Linear Regression as my base model. I will explain why and show the result based on Linear Regression.

In sklearn library in python, we can use the algorithms by calling the regressor classes directly. Logistic Regression requires to set a threshold value to transform target to 0 or 1, then computes the probability of 0 or 1. But it is not reasonable to set any threshold. For example, when I set ”chance of admit $>$ 0.73” as threshold, 230/500 sample is classified to admit, 270/500 sample is classified to not admit. From our logic thinking, does only chance greater than 0.73 can be defined as high possibility to be admitted? When I set the threshold as 0.5, 461/500 will be classified as admit, which means our data sample is super imbalanced. Therefore, Logistic Regression does not suite this case.

Let us consider Polynomial Regression. At the first, I do not know the features are linear related with the target or not. Linear relation is the simplest relation, we doubt it may

not fit our data properly. I tried Polynomial Regression. The only thing we should be careful is that Polynomial Regression may lead to overfitting as its flexible curve. So I observe the learning curve at the same time of adjusting the highest fitting power.

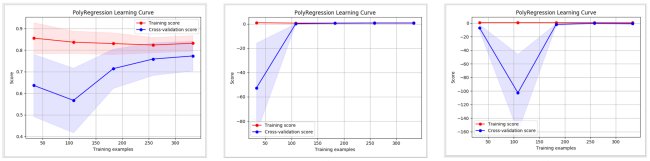
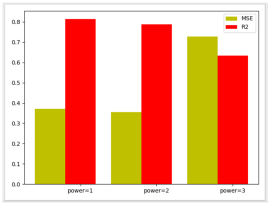


Fig. 12. Learning Curve of Polynomial Regression when (a) power=1; (b) power=2; (c) power=3



	estimator	MSE	R2
0	power=1	0.003705	0.813122
1	power=2	0.003548	0.787367
2	power=3	0.007279	0.632781

Fig. 13. Comparison of MSE and r2 scores when power=1, 2, 3

It is very obvious that when power increases, the performance reduces. When power $\geq$ 3, the curve is overfitted. If power $=$ 1, polynomial regression is linear regression. It indicates that the relation between features and target is tend to be linear.

We call sklearn.linear_model.LinearRegression to generate linear model. I use score function to get the accuracy of training and validation data set and use cross-validation to get a fair r2 score. The result is showed as follows:

```
**************LinearRegression***************
Parameters of model :
 LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
Predict:
 [0.91457473 0.79518127 0.57265986 0.70736968 0.81588282 0.86286561
 0.47459746 0.64850923 0.82378728 0.88741498 0.72193204 0.72589118
 0.65632227 0.93677168 0.8241518  0.58979177 0.83931942 0.59727295
 0.53339576 0.57155958 0.66548168 0.55305833 0.72232308 0.79500804
 0.78027648 0.60248654 0.94840363 0.84741471 0.62777011 0.74343096
 0.55533035 0.73004034 0.54474225 0.86116288 0.65713016 0.7371816
 0.55423839 0.95718977 0.64364267 0.71057279 0.97836982 0.57495143
 0.67075391 0.85830422 0.94112903 0.57793762 0.9583926  0.83902765
 0.79591651 0.92570648 0.88085969 0.56366238 0.70359711 0.52658929
 0.9536427  0.59746814 0.95608396 0.73916386 0.66256982 0.5812903
 0.62950759 0.68031188 0.59896721 0.59203806 0.44085868 0.58866369
 0.8667547  0.89783086 0.65831087 0.70667392 0.6176818  0.78587721
 0.69152566 0.56271819 0.5542953  0.65804583 0.84627224 0.86373777
 0.53729574 0.63142139 0.76958036 0.84812916 0.61693172 0.8471071
 0.73411583 0.6668525  0.60444455 0.73875671 0.78899999 0.66320147
 0.7428225  0.90802002 0.91576583 0.65856489 0.77694407 0.43563138
 0.68664259 0.78598826 0.73469446 0.64865736]
Training data accuracy in model is: 0.8210671369321554
Validation data accuracy in model is 0.8188432567829629
Mean squared error of test is : 0.0037046553907804097
R2 score: 0.81 (+/- 0.16)
```

Fig. 14. Scores of Linear Regression

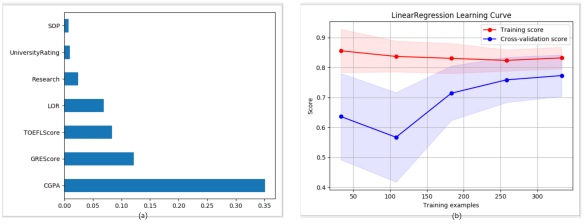


Fig. 15. (a) Weights of each feature in Linear Regression; (b) Learning curve of Linear Regression

In conclusion, for Linear Regression, the r2 score is 0.81 ± 0.16, MSE is 0.0037. This model does not overfitting.

D. Model training and testing

The most common used regression models are Linear Regression, Logistic Regression, Polynomial Regression, Ridge Regression, Lasso Regression, ElasticNet Regression [7]. There are solid statistic foundation supporting them. According to above subsection, we believe that this data relationship is more likely linear. Polynomial Regression is as the same as Linear Regression when power $=$ 1. Logistic Regression requires to transfer "chance of admit" to 0 or 1. Therefore we do not consider these two methods any more. The mathematics Theories behind Ridge Regression, Lasso Regression and ElasticNet Regression are similar, taking the correlation between independent variables into account, only the regularization methods are different. So we only pick Ridge Regression to represent this kind of algorithm. Except these easy interpreted algorithms, Neural Network is also an important method. Although unclear theory support about its undecidable performance, it is still an approach to predict the target. Random Forest Regression is unique in machine learning by its bagging idea.

Therefore, I choose Linear Regression, Ridge Regression, Multi-layer Perceptron (MLP) Regression, Random Forest Regression to train my model. Because I have analyzed Linear Regression in section III, I will give details about other methods and the corresponded training results respectively.

1) Ridge Regression: Ridge Regression is the development of Linear Regression. As we know, Linear Regression indicates the linear relation between independent variables and dependent variable. However, in some practical cases, independent variables are also correlated, this must affect the result. For example, in this case, "GREScore", "TOEFLScore" and "CGPA" are highly correlated according to Figure 9. It may because the students who get high "GREScore" work harder than others, so hardworking students generally achieve higher scores in these three items. This phenomenon is called collinearity. Linear equation can be decomposed into bias and variance. Although least square estimation are unbiased, their variances are large, which makes observed value far from the real value. In order to reduce the negative effect bringing by collinearity of independent variables, Ridge Regression

introduces shrinkage parameter λ to regression equation on the basis of Linear Regression.

$$\min \underbrace{||y - \mathbf{X}\beta||^2}_{Loss} + \lambda \underbrace{||\beta||^2}_{Penalty}$$

where $\mathbf{X}$ is the variables matrix, y is true value, β is the coefficient. λ is the key point to shrink the parameter β to have a very low variance but not zero. In Sklearn, Linear Regression library, this method can be called directly. alpha is a main hyperparameter in this function. I used Grid Search to search the best alpha ranging from 0.1 to 1.0. Then I found 0.1 is the best value when other hyperparametes are default. The result of this model is show as follows:

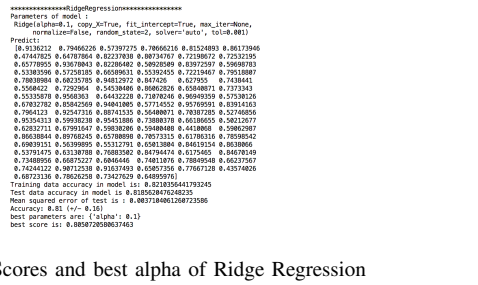


Fig. 16. Scores and best alpha of Ridge Regression

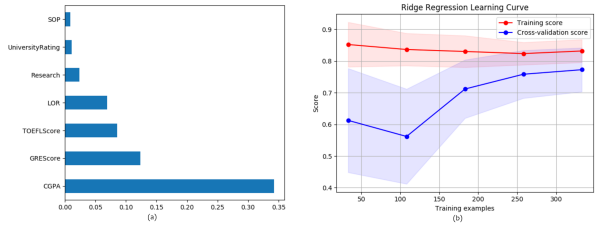


Fig. 17. (a) Weights of each feature in Ridge Regression; (b) Learning curve of Ridge Regression

2) Multi-layer Perceptron Regression (MLP): MLP is a supervised learning tool. It builds a non-linear learning function $f(\cdot) : \mathbf{R}^m \longrightarrow \mathbf{R}^o$ by learning from data, where m is the number of input features, o is the number of output variables. The principle can be explained as Figure 18 [8].

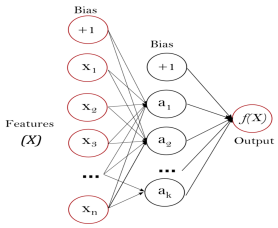


Fig. 18. One hidden layer in MLP.

In this case, we only have one output. This method is imitating human neuron system. Between input and output layers, there can be multi hidden layers. Except input and output node, every node stands for a neuro to train data by using a activation function. The training technique is called backpropagation, which can adjust weights of variables through learning until getting the minimal bias. There are four common activation functions [9].

$$Linear : A = cx; \quad Sigmoid : A = \frac{1}{1 + e^{-x}}$$

$$Tanh : \frac{2}{1 + e^{-2x}} - 1; \quad Relu : A = \max(0, x)$$

I choose *tanh(x)* as the activation function. Except this, the number of hidden layers is an important hyperparameter. After Grid Search, 8 hidden layers is the best. The result is show as follows:

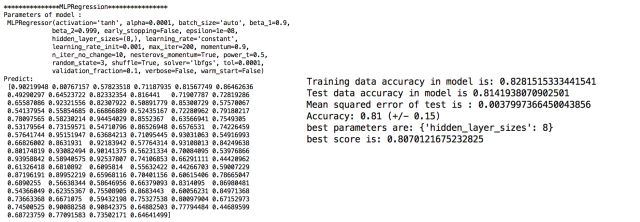


Fig. 19. Scores and best value of hidden layers layer in MLP.

3) Random Forest Regression: Random Forest algorithm is developed from Decision Tree. Decision Tree is a tree-like structure, every branch represents the outcome of an event, and every leave represents the label of one class. It is usually used in classification problem, but it can also be used to predict the possibility. However, with the deeper and deeper irregular pattern Decision Tree explores, it is very easy to cause overfitting, such as low bias but high variance. In order to avoid overtraining, Random Forest Regression was proposed by Leo Breiman [10]. Its main idea is to build several random decision trees and bagging these trees. This method benefits the case containing many variables which abstracts advantages of all single decision trees. To Random Forest Regression function in Sklearn, n-estimators is the main hyperparameter, which means the number of trees in forest. Because there are not too many independent variables, I make n-estimators = 20. The result is showed as follows.

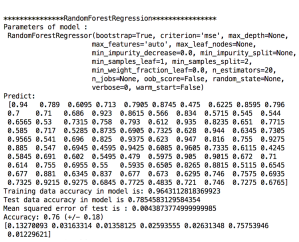


Fig. 20. Scores of Random Forest Regression.

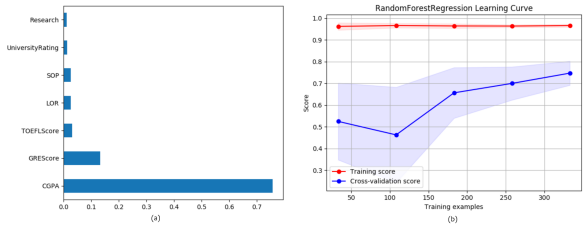


Fig. 21. (a) Weights of each feature in Random Forest Regression; (b) Learning curve of Random Forest Regression

V. Model ensemble and result analysis

We have got prediction results, how to evaluate our model? I will introduce two most important regression metrics: r2-score and MSE.

A. Model evaluation metrics

As I mentioned before, prediction error can be decomposed to two parts, bias and variants. Expected squared error at a point x is [11]:

$$\begin{aligned} Err(x) &= E[(\mathbf{Y} - \hat{f}(x))^2] \\ &= (E[\hat{f}(x)] - f(x))^2 + E[(\hat{f}(x) - E[\hat{(f)}(x))^2] + \sigma_e^2 \\ &= \text{Bias}^2 + \text{Variance}^2 + \text{Irreducible Error} \end{aligned}$$

Err(x) is the sum of Bias, variance and the irreducible error. Irreducible error is the noise in our data and uncontrolled. Bias means the difference between true value y and average of predicted y. High bias implies predicted data are very discrete. In other words, our model pays little attention to training set. So, it always leads to high error in both train and test data. Variance means how true value is spread out from average of predicted value. Simply to say, high variance means that model pays too much attention on training data so as to overtraining. In other words, this model performs well on training data but bad on test data. Our goal is to minimize both bias and variances.

- MSE (mean square error) represents the difference between predicted value and actual value. In Wikipedia [12], it is defined as the average squared difference between the estimated values and the actual value.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

  where $\hat{Y}_i$ is the predicted value, Y_i is the true value. It is never negative. Undoubtedly, the lower the value MSE is the better. MSE equals to 0 means the model is perfect.
- r2-score is also called Coefficient of determination, it means how well our model is correlated to actual data. In Wikipedia [13], it is defined as "r2 is the proportion of the variance in the dependent variable that is predictable from the independent variable(s)". In math, it is defined as "(total variance explained by model)/total variance". It ranges from 0 to 1. We hope more variance can be explained by our model, so the higher the score of r2 is the better. r2 equals to 100% means no variance.

$$\text{The total sum of squares } SS_{tot} = \sum_i (y_i - \bar{y})^2$$

$$\text{The sum of squares of residuals } SS_{res} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

$$\text{Coefficient determination } R^2 = 1 - \frac{SS_{res}}{SS_{tot}}$$

Put all results together, we can conclude that the accuracy of Random Forest Regression performs is lower than other algorithms'. It is very obvious that it performs super well on training data but bad on test data according to its learning curve. Other regressors perform similarly and are not overfitted obviously. The r2 score are around 80% and MSE values are around 0.004. Overall, their generalize capacity are proper. In order to improve their performance, I introduced ensemble method to combine these regressors by stacking.

B. Model Ensemble

There are four popular ensemble methods in regression problem: Averaging, Bagging, Boosting and Stacking [14].

- Averaging: Assume we have three models, every model gets a prediction y, we take the average value $\bar{y}$ of them as the final prediction. Weighted Average is the development of averaging as it assigns weight to each model to get the final result. It is more reliable than simple averaging.
- Bagging: Its idea is combining the prediction results of different models such as decision trees to get a generalized result. Random Forest is a classic example. It builds *n* decision trees based on random subsets and then combines them. It requires all of models are strong so that achieves stronger model finally.
- Boosting: Different from Bagging, Boosting technique pays more attention to incorrect results. It is a sequential process. When a model makes mistakes in data, the next model will give a higher weight to errors and correct them. After several iterations, the final model will be strong although each model does not work well. It even requires every sub-model is weak, every model boosts the performance through ensemble. AdaBoosting is a popular example.
- stacking: In my project, I use this method to combine my basic models. Its main idea is to use prediction results produced by basic models as new features to train a new model. This is explained by Figure 22.

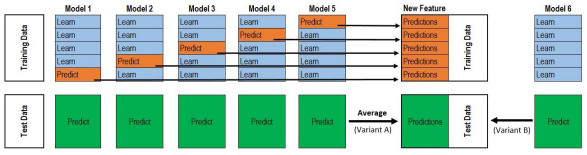


Fig. 22. The process of stacking

For example, we have 5 first lever models and 1 second level model, 1000 training data with 7 features. Firstly, separate 1000 sample data to 900 training data and 100 test data. Secondly, every basic model picks 700 data from 900 data to train then predicts the remained 200 data get 200 prediction result. The shape of new data set becomes 5×200 (5 new features for each data). Put these data to 6th model to get the final model, which predicts 100 test data we separated before to get the accuracy score.

The advantage is that it can combine the advantages of all models we trained before. It can contain several levels, every previous model is taken as input of the next model. I only built two levels. In order to avoid overfitting and abstract more comprehensive information of original data, I use Linear Regression, Ridge Regression, MLP Regression, Random Forest Regression, Gradient Boosting Regression and AdaBoosting Regresson as first level model, then apply Ridge Regression to combine them to produce final prediction. The procedure and result are show as follows:

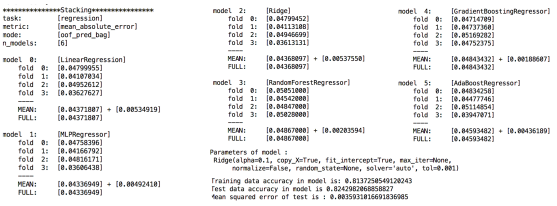
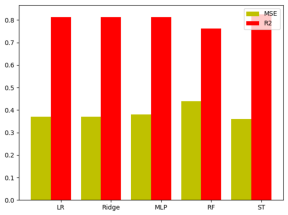


Fig. 23. Stacking result

C. Result anlysis

The final results and comparison between models on MSE and r2-score are as bellow:



	estimator	MSE	R2
0	LR	0.003705	0.813122
1	Ridge	0.003710	0.813112
2	MLP	0.003800	0.812415
3	RF	0.004387	0.761442
4	ST	0.003593	0.824298

Fig. 24. Result comparison

In this project, I paid attention to six aspects: given weights to every feature, learning curve of training data and test data, training data score, test data score, MSE and r2-score.

Considering to the weights Linear Regression, Ridge Regression and Random Forest Regression assign to every feature, all of them give highest weight to "CGPA" but lower weights to "SOP", "UniversityRating" and "GREScore". This is in line with previous expectation. However, Linear Regression and Ridge Regression assign almost same weights to corresponded features, Random Forest weakens the effect of "TOFELScore" and "GREScore" sharply. The judgement to the importance of every feature from Random Forest Regression may lead to the lower r2-score.

Considering to learning curve I plotted and the accuracy score of training and test data, Linear Regression and Ridge Regression achieve almost the same. The accuracy score here means r2-score, the difference from r2-score in the final result is that accuracy score are based on separate training and test data, r2-score is got by cross validation based on whole data sample. Comparing training and test score helps us to see if the model is overfitted or underfitted. If training score is far higher than test's, this model is overfitted. If test score is far higher than training's, it is underfitted, which means we need more data to train model. The training and test score are similar in all methods except Random Forest Regression. Its training score is approximately 96.4% but test score is only about 78.5%. It is also very obvious to see from the learning curve that this model is overfitted.

Our goal is to minimize MSE and maximize r2-score. Because of the differences of different algorithms, I used stacking to combine them. Before stacking, Linear and Ridge Regression got almost same score, 0.0037 in MSE and 0.813 in r2-score. Random Forest Regression performs the worst, only get 0.76 in r2-score but has the highest MSE 0.004. The performance of MLP stays in the middle. After stacking, according to Figure 24, MSE is reduced to 0.0035 and r2-score is improved by 0.01 to 0.824. Although the performance is not improved too much, but still makes progress to the final prediction.

VI. Conclusion

My project mainly includes three parts: data preprocessing, base model selection, modeling and stacking. Because the original data is completed and clean without too many features, most energy I spent on my project is the base model selection and stacking. Selecting base model is important so as to I save more energy in result analysis and have base line to compare with following models. I chose Linear Regression as my first model, the truth proves that I do not need to set any hyperparameter but get a better model than average level. It is also a important step to know the features of original data, which is benefited to choose following models and analysis metrics. Stacking is what I innovate in problem. Because my previous models are good, I want to take full advantages of them, stacking is ideal for this case. The a little bit of progress proves my guess but not too much.

I think my final model can be optimized to a certain degree. There are a few factors affecting my modelling. Firstly, except Linear Regression, all regression contains many hyperparameters. Every hyperparameter has its own usage and meaning in Sklearn library. Under the situation that I do not know which algorithm suites my project best, so I only adjusted some hyperparameters. Exploring deeper about them can help improve single model. Secondly, the principle of stacking is

simple but its usage is not so easy. How to choose the first level and second level models? What kind of regressors combination can achieve better result? What is the precondition of data and basic models before stacking? How many levels should be built? Thirdly, the original data set is not accurate, this is an important reason why the accuracy can not be improved more. As we know, "Graduation admission" is true or false problem. Where does the possibility infers from? After reading some materials, these possibility of chance of admission are from appliers. They guess the chance. This is not accurate, which may cause a lot noise in modeling. If we treat this set of data as a classification problem, the bias will be smaller although I think possibility of event happening is more meaningful.

REFERENCES

[1] N. Datta. (2019) Graduate admissions. [Online]. Available: https://www.kaggle.com/nitindatta/graduate-admissions

[2] V. Garnepudi. (2019) Admission chances. [Online]. Available: https://www.kaggle.com/venky73/admission-chances

[3] A. Tiha. (2019) Admission prediction (mean squared error - 0.0013). [Online]. Available: https://www.kaggle.com/anjanatiha/admission-prediction-mean-squared-error-0-0013

[4] suhas maddali. (2018) Chance of admission (deep neural networks). [Online]. Available: https://www.kaggle.com/suhasmaddali007/chance-of-admission-deep-neural-networks

[5] M. Peixeiro. (2018) Linear regression understanding the theory. [Online]. Available: https://towardsdatascience.com/linear-regression-understanding-the-theory-7e53ac2831b5

[6] F. Nerdy. What is r squared and negative r squared. [Online]. Available: http://www.fairlynerdy.com/what-is-r-squared/

[7] S. RAY. (2015) 7 regression techniques you should know! [Online]. Available: https://www.analyticsvidhya.com/blog/2015/08/comprehensive-guide-regression/

[8] Sklearn. Mlpregressor homepage in sklearn. [Online]. Available: https://scikit-learn.org/stable/modules/neural-networks-supervised.html

[9] A. S. V. (2017) Understanding activation functions in neural networks. [Online]. Available: https://medium.com/the-theory-of-everything/understanding-activation-functions-in-neural-networks-9491262884e0

[10] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[11] S. Singh. (2018) Understanding the bias-variance tradeoff. [Online]. Available: https://towardsdatascience.com/understanding-the-bias-variance-tradeoff-165e6942b229

[12] Wikipedia2. Mean squared error. [Online]. Available: https://en.wikipedia.org/wiki/Mean squared error

[13] Wikipedia. Coefficient of determination. [Online]. Available: https://en.wikipedia.org/wiki/Coefficient of determination

[14] A. SINGH. (2018) A comprehensive guide to ensemble learning (with python codes). [Online]. Available: https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-for-ensemble-models/