CS273P Project 1: Predict Rainfall

TEAM MEMBERS AND WORK DISTRIBUTION

Team Name: EL PSY CONGROO (Final Kaggle Score: 3.41391)

Team Members	Student ID	Responsibilities
Po-Ren Pan	10202449	Exploratory Data Analysis, Random Forest, Stacked Random Forest
Chun-Kai Chan	41525034	K-Nearest Neighbors, Random K-Nearest Neighbors
Yi-Dar Liu	32845675	Exploratory Data Analysis, Linear Regression, Simple Ensemble, Stacked Linear Regression

First, we discussed the problems to solve, and then each was responsible for coding, data analysis, model selection, and writing of a specific technique. Afterwards we shared ideas and results to find ways to improve model performance.

INTRODUCTION

In this project we employ a dataset of 100,000 data points with 14 anonymous features (denoted by Xi, i = 1, 2, ... , 14) to predict the amount of rainfall (denoted by Y) for 100,000 locations. The performance is evaluated using the Mean Square Error (MSE) of test data by Kaggle. We would like to tackle three problems:

1. Technique selection: Which technique works best for the dataset?
2. Data manipulation (including selection and transformation): Which data should be used to train our model? Will transformed data produce better results?
3. Model selection (including feature and hyperparameter selection): Which model has the largest predictive power?

Each problem is discussed in detail below.

Technique selection

In this project, we explore three kinds of learners: Random Forest (RF), K-Nearest Neighbors (KNN), and Linear Regression (LR). Since the techniques are based on different assumptions and rationales (summarized in the table below), their predictions should be uncorrelated. Therefore, they should be suitable for the ensemble method.

	Tool	Assumption and Rationale	Advantage and Disadvantage
RF	sklearn	Assumption: Target value is based on a series of branching on feature thresholds (decision tree) Rationale: Learn a collection of models and combine them to get a more accurate and stable prediction	Advantage: • Use bagging and randomization to reduce memorization effect and variance • Computationally fast Disadvantage: • Randomly sample data for training • It is hard to interpret because RF is an ensemble algorithm of decision trees
KNN	sklearn	Assumption: Close features lead to close target values Rationale: Use the average value of k nearest neighbors as the output	Advantage: • Predicted value will always be within the domain of target value, so it will always be nonnegative. Disadvantage: • Do not work well in high dimensions • Computing distances in large data sets is slow (e.g. 500 runs take 30 minutes)
LR	PHStat mltools	Assumption: There is a linear relationship between features (could be quadratic or interaction) and target value. The errors are normally distributed and have equal variance. Rationale: Minimize MSE to optimize parameters	Advantage: • Computationally fast Disadvantage: • Coefficients are sensitive to data values, especially outliers • If the relationship is nonlinear, the predictive power is low. • Cannot use bagging • May predict negative values. In this case, we assign zero to the target value.

Data manipulation (including selection and transformation)

The following three different ways can be used to select training data, and the optimal way may depend on the technique used.

	Advantage	Disadvantage
Use the whole data set	Use the full information	• Data may contain errors, which will affect model performance • Computation is slow
Randomly select subsets of data and repeat several times	• Excepted to eliminate outliers in a natural way • Computation is fast	Do no use the full information
Eliminate outliers before employing the above two ways	Models have better in-sample performance	• Definition of an outlier is subjective • Outliers may not be pure errors • May fail to predict extreme cases

Besides, features and target value can be transformed by scaling or being replaced by their derivatives (functions of variables). We would like to know if transformed data generate better results.

Model selection (including feature and hyperparameter selection)

The following three options can be used to select features, and the optimal way may depend on the technique used.

	Advantage	Disadvantage
Include all features	Use the all information	• Features may not be independent • Computation is slow • Some features may have no influence on target value and thus could hurt a model's out-of-sample performance
Include only a subset of features	Can focus on the most important features when optimizing parameters	• Judging feature importance is subjective • Other features may have subtle influence on target value

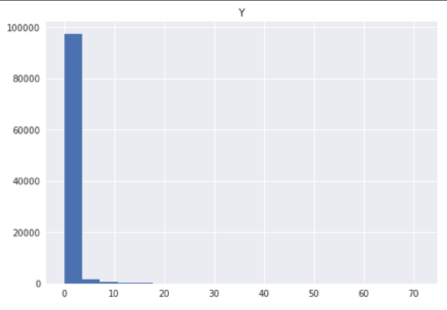
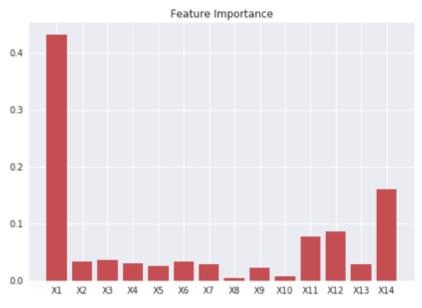
Hyperparameter selection will be discussed in the following individual technique sections.
Notation: [i] means ith reference in APPENDIX I.

RANDOM FOREST (RF)

First, we include all data without any manipulation to train our model. However, it does not perform well in 5-fold cross-validation. We guess the poor performance could be attributed to large variations in feature values. For example, X5 and X6's standard deviations are 3952 and 1738, respectively. (Please refer to APPENDIX II for all features' descriptive statistics.) Therefore, we take logarithm of the variables to reduce variance. Next step is deciding the hyperparameters. Based on the 5-fold cross-validation results, we select number of features = 14 and max depth = 7.

Figure 1 shows that the majority of Y values in training data are close to zero. We worry that if the whole data set is used to train our model, Y values could be underestimated. Accordingly, we randomly select 10,000 data points from each of the two subsets "$Y < 1$" and "$Y \geq 1$" to train our model. Unfortunately, the performance is worse than before (Kaggle score: 4.31992). It suggests that the actual distribution of Y should be right-skewed, so naturally most of training data have small Y values.

Another approach is to select important features before training. The intuition is to use weighted impurity decrease for all nodes in the decision trees [3]. (Please refer to APPENDIX III for the formula of feature importance $Imp(X_m)$.) Based on the result of sklearn's feature_importances_ (shown in Figure 2), we choose the top 50% features, namely X1, X3, X7, X11, X12, X13, X14, to re-train our model. However, the new model's performance does not improve.

Figure 1: Y's Histogram	Figure 2: Feature Importance in RF
	

K-NEAREST NEIGHBORS (KNN)

Initially, the standard KNN is used. One of the critical parameters is k, the number of nearest neighbors used for training. To find k, we use 5-fold cross-validation and calculate cross-validation error (measured in terms of average MSE) for each k. As Figure 3 shows, the ideal k (where average MSE is minimized) is 50. Under this setting, our model obtains a score of 3.56363 on Kaggle. Next, we try to improve the performance of the KNN model by using the Random K-Nearest Neighbors (RKNN) model, which is described below.

Figure 3: Cross-Validation Error as a Function of k

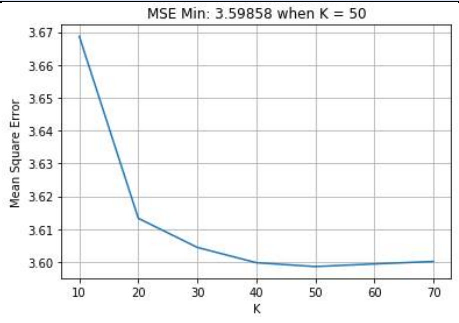
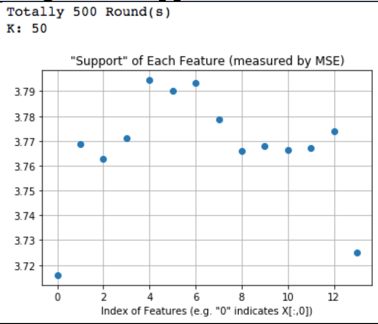


Figure 4: Supports of Features



<u>Improvement: Random K-Nearest Neighbors (RKNN)</u>

To improve the prediction accuracy, we try to decide a subset of features that have the most predictive power. A randomized method for feature selection [4] is used. The method divides the selection algorithm into 2 phases: (1) calculate the feature support and (2) select top important features. During this process, we use k = 50.

Feature Support - Bidirectional Voting

Feature support means the classification capability of a feature. To determine the *support* of a feature, we randomly select a subset of features, X_m, which has m features. Then we use the KNN classifier to train the model and get the accuracy (measured in terms of MSE) for each X_m. The steps above are repeated *r* times. After all the iterations, each feature will appear in some KNN classifiers. The mean accuracy of these classifiers, called *support*, is a measure of the feature's relevance with the outcome. The lower support a feature has, the more it is relevant to the outcome.

RKNN Feature Selection

After getting the *supports* of all the features, we carry out feature selection. There are two ways. The first is *direct selection*, which simply selects high ranking features. However, we consider it too aggressive. Alternatively, we adopt a conservative approach described in [4]. First, we iterate several rounds of direction selection, and a fixed number of features are dropped in each round, in other words, a subset of features is selected for training. We train the KNN model on this subset. Then we plot model accuracy against decreasing number of features to find the optimal number of features. Once the number is determined, we use 5-fold cross-validation to find the optimal k for these features. Finally, we predict the output with the optimal number of features coupled with the optimal k.

Result

In the *feature support* phase, we run KNN $r = 500$ times with $m = 4 = \sqrt{14}$ features, which is recommended by [4]. As Figure 4 shows (here x-axis = i means Xi+1, e.g. x-axis = 0 means X1), X1 and X14 have relatively high relevance with the output, while X5, X6, and X7 have the lowest relevance with the output.

In the second phase, we utilize the *RKNN feature selection* algorithm. The result is shown in Figure 5. It indicates that eliminating 3 features has the best accuracy of predicting the validation data. The final selected features are X1, X2, X3, X4, X8, X9, X10, X11, X12, X13, X14.

Figure 5: Cross-Validation Error as a Function of Number of Features Eliminated

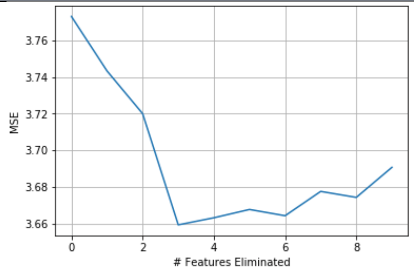
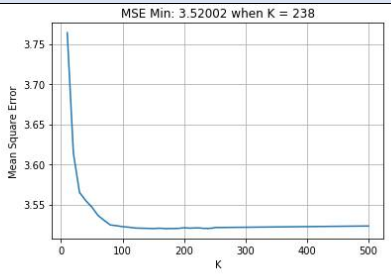


Figure 6: Cross-Validation Error as a Function of k after Feature Selection



Based on this subset of features, we utilize 5-fold cross-validation to decide the optimal k. The relationship between k and cross-validation error is shown in Figure 6, which indicates that the best prediction occurs at k = 238. It is a large number. The positive side is that by averaging so many data points, we can reduce the impact of outliers and obtain better prediction results. The negative side is that it may result in underfitting (low variance and high bias). However, the problem is alleviated by the sample size we have (100,000 data points). Using k = 238, we get new Kaggle score = 3.50821. It is a substantial improvement from the standard KNN model (Kaggle score = 3.56363). In addition, it is the highest score we achieve with a single model.

LINEAR REGRESSION (LR)

First, we include all training data in our analysis. And then we exclude outliers from our sample and evaluate if they have significant impact on our result. APPENDIX I summarizes key characteristics of features. We notice that some variables are strongly correlated, as the following table shows (ρ: correlation coefficient). It implies collinearity among these variables.

Strong Positive Correlation ($\rho > 0.75$)	**Strong Negative Correlation ($\rho < -0.75$)**
{X2, X3, X4}, {X5, X6}, {X9, X10}, {X11, X12}	{X2, X3, X4} and {X9, X10}

Collinear variables do not provide extra information, and their coefficients may fluctuate drastically, resulting in an unstable model. We use variance inflationary factor (VIF) to evaluate the collinearity among the variables. If a variable can be explained by other variables, then it will have a large VIF. Statistical package PHStat is used to calculate VIF. We refer to Snee [1] and drop variables with VIF more than 5. Finally, the remaining 9 variables are: X1, X4, X5, X7, X8, X11, X12, X13, X14. As we expected, highly correlated variables are removed from the feature pool.

And then we need to decide which subset of the non-collinear features should be put in our model. Our heuristic is "parsimony": select the model with the fewest independent variables that can predict the dependent variable adequately. Simpler models are less prone to collinearity and over-fitting problems. We use two methods to select features: Statistical Inference and Cross-Validation.

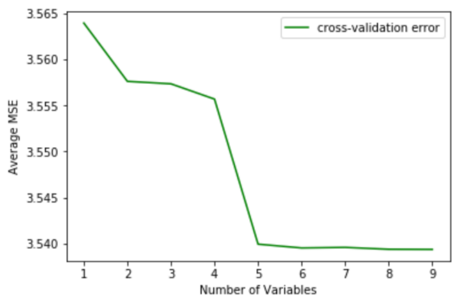
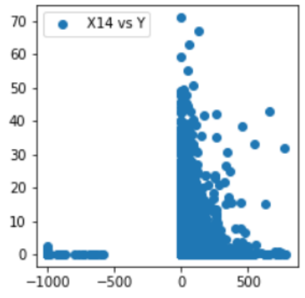
Method 1: Statistical Inference (adjusted r^2, C_p, and *p*-value)

We consider three statistics: adjusted r^2, C_p, and *p*-value. Their formulas are listed in APPENDIX III. r^2 measures the proportion of the variation in Y that is explained by the independent variables. adjusted r^2 takes into account the number of independent variables in the model, so it is more suitable for selecting models with varying number of features. C_p measures the differences between a fitted regression model and a true model, along with random error. When a regression model with k independent variables contains only random differences from a true model, the mean value of C_p is k + 1. It is advised to select models whose C_p is close to or less than k + 1 [2]. *p*-value for an independent variable is the probability that, when there is no linear relationship between the independent variable and the dependent variable, the calculated coefficient would be of greater magnitude than the actual observed result. Low *p*-value (e.g. < 0.1) indicates that the relationship is statistically significant.

First, we use PHStat to execute best-subsets regression to generate all possible models. Because PHStat can only work on at most 7 features, we look at the full model and drop two variables with the highest *p*-values, X4 (*p*-value = 0.8793) and X5 (*p*-value = 0.7046). We find that 28 models among 127 have C_p values equal to or less than k + 1. Besides, their adjusted r^2 are quite close (maximum and minimum differs by merely 0.002). Therefore, it is hard to select models based on the two criteria. It is worth noticing that X1 is included in all 28 models.

Alternatively, we use *p*-value to select features. We demand that the relationship between a feature and the target value be stable regardless of the data, so we randomly select a subset of size 10,000 from the training data, run regression on all 9 variables, record their *p*-values, and repeat the process multiple times. If the sign of the coefficient (positive/negative) is consistent and the *p*-value also remains below some threshold (0.5 in our case), we include the variable in our model. After 10 runs, only 5 variables meet the standards: X1, X7, X11, X12, X14. We look back at the best-subsets results and find that the model's C_p value is 7, close to the ideal value of 6, so we select the model.

In order to ensure that the selected variables are sufficient to explain the data, we construct models by adding variables one at a time in the increasing order of average *p*-values: X1, X7, X11, X12, X14, X4, X8, X13, X5. And then we use 5-fold cross-validation and scaled data to verify that the five-variable model is neither underfitted nor overfitted. Figure 7 shows cross-validation error (measured in terms of average MSE) as a function of number of variables. As our previous regression analysis shows, if the model has included X1, X7, X11, X12, X14, other variables only slightly improve performance. Moreover, we notice that although X14's *p*-value is not small, including X14 in the model causes the error to plunge, so we suspect that the relationship between X14 and Y is non-linear. Figure 8 shows the scatter plot of X14 (x-axis) and Y (y-axis). It is apparent that X14 are clustered into two groups: "≥ 0" group and "< -500" group. Let us call the former "Case A" and the latter "Case B."

Figure 7: Cross-Validation Error as a Function of Number of Variables	**Figure 8: Scatterplot of X14 and Y**
	

The relationship between Y and the other four features, namely X1, X7, X11, X12, may be different in the two cases, so we divide the training data into two subsets based on X14 and train two learners on each subset. We find that after the separation, X14's coefficient becomes more insignificant (i.e. X14 has a higher *p*-value); on the contrary, the other four variables' *p*-values substantially decrease (all of them become less than 0.1). We drop X14 in each case and re-run the model, finding that the results remain almost unchanged, so finally we choose the two-case model with X1, X7, X11, X12 as explanatory variables. The following is the final model built using this method: (based on scaled X values)
Case A ($X14 \geq 0$): $Y = 0.4598 - 0.3704*X1 - 0.0893*X7 - 0.0335*X11 + 0.0688*X12$
Case B ($X14 < -500$): $Y = 0.0439 + 0.0249*X1 - 0.0218*X7 - 0.0367*X11 + 0.0633*X12$

Please note that the sign of X1's coefficient is different between the two cases. Besides, the intercept of Case A is much larger than Case B, which coincides with the previous result that X14's coefficient is positive. Finally, we perform a complete analysis to verify that the model satisfies the regression assumptions. None of the residual plots versus the independent variables reveal apparent patterns. In addition, histograms of the residuals show that they are almost normally distributed. Plots of the residuals versus the predicted values of Y does not show evidence of unequal variance. Since the data points correspond to different locations instead of time points, we do not have to worry about autocorrelation. We also try adding quadratic terms (e.g. Xi^2) and exploring nonlinear relationships (e.g. log(Y+1), Y/Xi) but fail to improve the model. Thus, we conclude that the model is appropriate for predicting rainfall.

Next, we would like to know how outliers affect our results. According to APPENDIX II, outliers do exist. For example, 99.1% of X5 values lie between 0 and 20,000, but still 0.2% are more than 30,000. We do not know if the values result from rare natural events or human errors. If this is the latter case, we expect that removing the errors ("noise") would help enhance the predictive power of our model. Our definition of an outlier is: away from the mean (μ) by more than 5 standard deviations (σ). If X is normally distributed, then the probability that $\left|\frac{X-\mu}{\sigma}\right| > 4.89$ is 0.000001. In a sample of 100,000 data points, it translates to an expected value of 0.000001*100,000 = 0.1, which is close to zero. The following table shows sample sizes before and after outliers are eliminated:

Case	Original Sample Size	Sample Size after Outliers are Eliminated	Number of Outliers
A (X14 ≥ 0)	99,731	96,509	3,222 (3.2%)
B (X14 < -500)	269	258	11 (4%)
Total	100,000	96,767	3,233 (3.2%)

Although the model displays a better in-sample performance (possibly because it does not need to predict extreme cases), the training error of the whole dataset is quite huge. It seems that the outliers play some role in determining the relationship between features and the target value, so we decide to keep them when training our model.

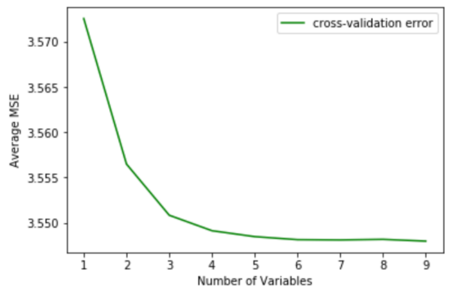
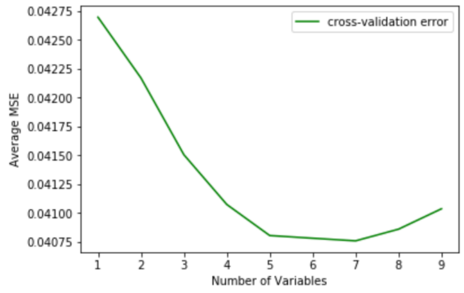
We do not use regularization in our regression model because:
1. Regularization adds penalty to all coefficients' magnitude regardless of their statistical significance. We care about the latter more than the former.
2. Quadratic terms do not improve model, so our model does not include higher-order terms. In this case, we do not think regularization is necessary.

Method 2: Cross-Validation (average MSE)

Another way to select models is based on cross-validation errors (measured in terms of average MSE). We still use the 9 non-collinear variables X1, X4, X5, X7, X8, X11, X12, X13, X14 to construct our models. Our goal is to find a threshold of the number of variables where cross-validation error does not improve when more variables are added. As with our previous cross-validation test, we use 5-fold cross-validation and add variables one at a time. If a variable can reduce cross-validation error more than others, we move it to the front of the feature list. Figure 9 and Figure 10 show the two cases' cross-validation results, respectively. Please note that both graphs are convex because the variables are added in the decreasing order of their contributions to error reduction. In Case A, once the 5 variables, X1, X14, X7, X12, X4, are included in the model, other variables do not provide extra boost to the performance. In Case B, once the 5 variables, X12, X11, X1, X8, X7, are included in the model, additional variables only slightly reduce error and may even make things worse (error increases when the number of variables is more 7).

The following is the final model built using this method: (based on scaled X values)
Case A ($X14 \geq 0$): $Y = 0.4598 - 0.2804*X1 - 0.0341*X4 - 0.0878*X7 + 0.0383*X12 + 0.1552*X14$
Case B ($X14 < -500$): $Y = 0.0439 + 0.0262*X1 - 0.0208*X7 - 0.0194*X8 - 0.0313*X11 + 0.0643*X12$

Since the model in Method 2 (Kaggle score = 3.53481) outperforms the model in Method 1 (Kaggle score = 3.53542), the model in Method 2 is used to create ensemble.

Figure 9: Cross-Validation Result of Case A	**Figure 10: Cross-Validation Result of Case B**
Variable Order: X1, X14, X7, X12, X4, X11, X5, X8, X13	Variable Order: X12, X11, X1, X8, X7, X13, X14, X5, X4
	

ENSEMBLE GENERATION

Finally, we combine the three learners and hope to get a better result. The unweighted average method and the stacking method are chosen as our ensemble algorithms. The unweighted average method is straightforward. That is, we average the predicted Y values as our final result. The stacking method is a two-step training scheme. First, we train our learners ("base learner") and generate predictions $\hat{Y}_{tr}$ and $\hat{Y}_{te}$ for training data and test data, respectively. We use $\hat{Y}_{tr}$ as training data of another model ("stacked learner"), and then use $\hat{Y}_{te}$ as input in the model to predict the final result. Two kinds of stacked learners are used: Linear Regression and Random Forest. Their results are shown below.

Method 1: Unweighted Average of RF, KNN/RKNN, and LR

Interestingly, despite some variation among our predicted values, the unweighted average method (Kaggle score = 3.49708/3.50226 if KNN/RKNN is used) outperforms any single model. It demonstrates the synergy of independent models.

Method 2: Stack RF, KNN/RKNN, and LR using Linear Regression

The resulting model is:
$Y = -0.0961 + 0.4712*Y_{rf} + 0.3803*Y_{knn} + 0.3806*Y_{lr}$ if KNN is used
$Y = -0.0124 + 0.3631*Y_{rf} + 0.4874*Y_{rknn} + 0.1926*Y_{lr}$ if RKNN is used
where Y_x is the target value predicted by base learner x

It is worth noticing that a base learner's coefficient is proportional to its individual Kaggle performance. We find that the linear regression stacking method (Kaggle score = 3.49415/3.50004 if KNN/RKNN is used) is even better than the unweighted average method, though only slightly.

Method 3: Stack RF, KNN, and LR using Random Forest

Although its in-sample performance is good (training MSE = 3.17037), its Kaggle score (3.50223) is worse than those of Method 1 and Method 2.

Method 4: Stack 5 RFs and LR using Random Forest

Due to RF's good performance, we try a stacked learner which primarily consists of different kinds of RFs. When choosing RF models, we conjecture that if they vary from low to high complexity, the combined performance may be better than a single optimized RF. The following 5 RFs coupled with LR are used to create the new stacked learner. Its Kaggle score is 3.4513, surpassing the previous four methods.

RF	1	2	3	4	5
Number of features	7	7	Default	5	Default
Max depth	5	12	Default	Default	8

Method 5: Stack 5 RFs, LR, and RKNN using Random Forest

We also try adding RKNN to Method 4 and setting max depth of the stacked Random Forest to be 5 to avoid overfitting. Its Kaggle score is 3.41391, which is the highest score we obtain so far.

CONCLUSION

We conclude our analysis by answering the questions described in INTRODUCTION:

1. **Which technique works best for the dataset?**

   The following table summarizes performances of all models. As can be seen from the table, in terms of Kaggle score, RKNN is the best individual model, and Method 5 is the best ensemble method. All ensemble methods perform better than individual learners. It demonstrates the benefit of combining independent learners.

Individual	Training Error	Kaggle Score	Ensemble	Training Error	Kaggle Score
RF	3.51812	3.51926	Method 1*	3.50566/3.50961	3.49708/3.50226
KNN	3.59858	3.56363	Method 2*	3.49779/3.50801	3.49415/3.50004
RKNN	3.52002	3.50821	Method 3	3.17037	3.50223
LR - Method 1	3.55410	3.53542	Method 4	2.97503	3.45143
LR - Method 2	3.53735	3.53481	Method 5	3.26001	3.41391

   *Left side uses KNN, and right side RKNN.

2. **Which data should be used to train our model? Will transformed data produce better results?**

   First, we use the whole dataset to train our models. We also try training our models on different subsets of data. For example, we train RF on a data set where $Y < 1$ and $Y \geq 1$ have equal weights, but the result is worse. On the contrary, dividing dataset by $X14 \geq 0$ and $X14 < -500$ improves statistical significance of features in LR. We also attempt to exclude outliers, but it does not result in meaningful improvement of the models' predictive power. Different ways of data transformation are tried. In the RF model we take logarithm of variables to reduce variance. We train the LR model on scaled data to obtain more stable coefficients.

3. **Which model has the largest predictive power?**

   Each technique selects quite different features:

RF	X1, X3, X7, X11, X12, X13, X14
KNN	X1, X2, X3, X4, X8, X9, X10, X11, X12, X13, X14
LR	Method 1: X1, X7, X11, X12, X14 Method 2: X1, X4, X7, X8, X11, X12, X14

   Nevertheless, all methods select X1, X11, X12, and X14. It shows that these four variables are the most important features in predicting Y.

   Each model's parameters are chosen as follows:

RF	Number of features = 14, max depth = 7
KNN	k = 238
LR	Case A ($X14 \geq 0$): $Y = 0.4598 - 0.2804*X1 - 0.0341*X4 - 0.0878*X7 + 0.0383*X12 + 0.1552*X14$ Case B ($X14 < -500$): $Y = 0.0439 + 0.0262*X1 - 0.0208*X7 - 0.0194*X8 - 0.0313*X11 + 0.0643*X12$

In this project we learn the trade-offs involved in machine learning. Computationally demanding methods like KNN can be powerful, but it also requires substantial running time. Besides, when a model becomes more complicated, it has lower training error, but it is also easier to be overfitted and produce bad out-of-sample result. We also learn that there is no one-size-fit-all technique in machine learning. In order to find a good model, one needs to try different combinations of techniques, data manipulation schemes, features, and hyperparameters. However, consolidating learners usually improves performance.

APPENDIX I: REFERENCES

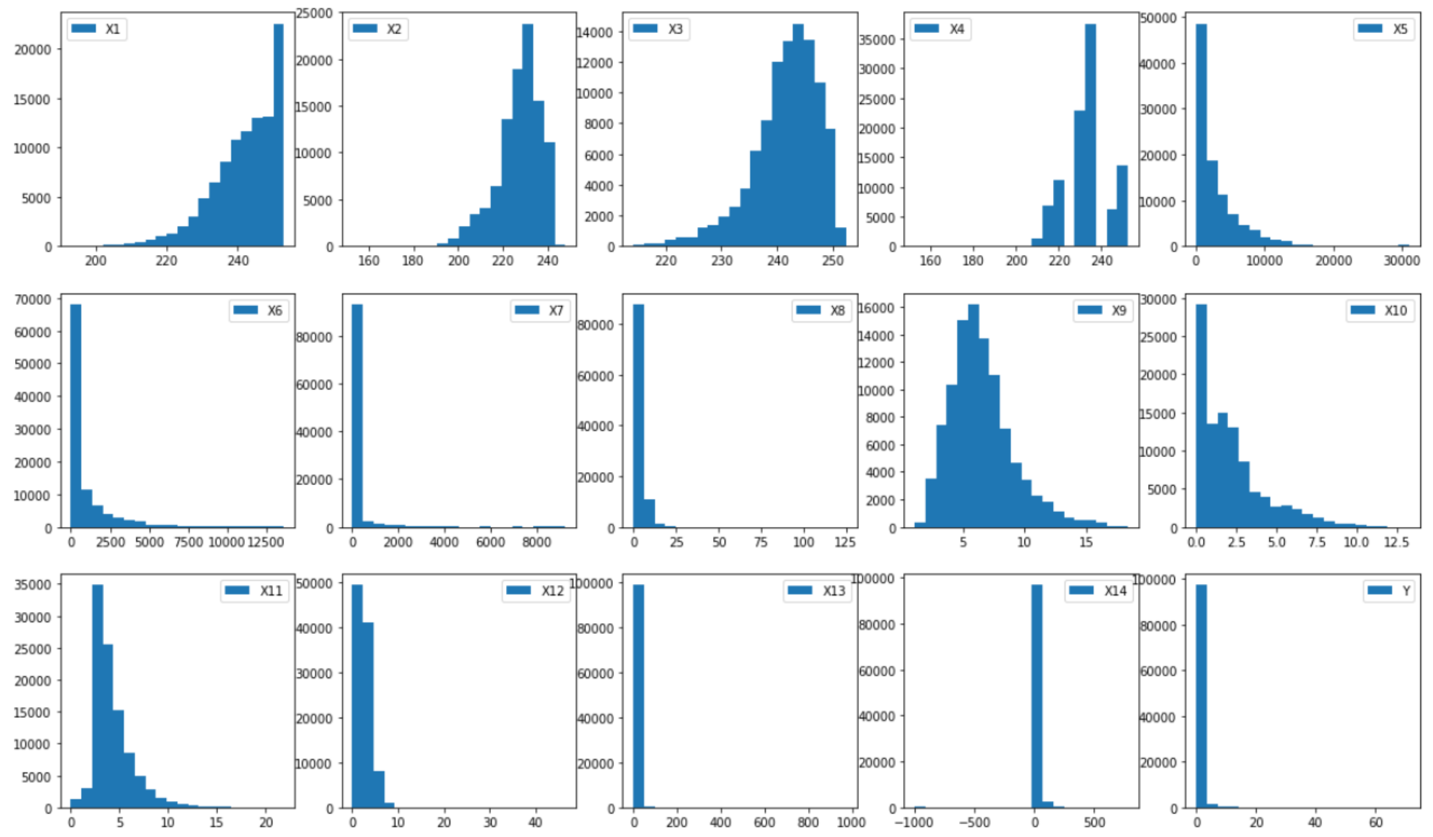
1. Snee, R. D., “Some Aspects of Nonorthogonal Data Analysis, Part I. Developing Prediction Equations,” Journal of Quality Technology, 5 (1973), 67–79.
2. Kutner, M., C. Nachtsheim, J. Neter, and W. Li, Applied Linear Statistical Models, 5th ed. (NewYork: McGraw-Hill/Irwin, 2005).
3. Gilles Louppe, Louis Wehenkel, Antonio Sutera and Pierre Geurts, “Understanding Variable Importances in Forest of Randomized Trees,” NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems, 1 (2013), 431-439.
4. Shengqiao L., E James H., Donald A A., “Random KNN Feature Selection – A Fast and Stable Alternative to Random Forests,” BMC Bioformatics 2011, 12:450.

APPENDIX II: EXPLORATORY DATA ANALYSIS RESULTS

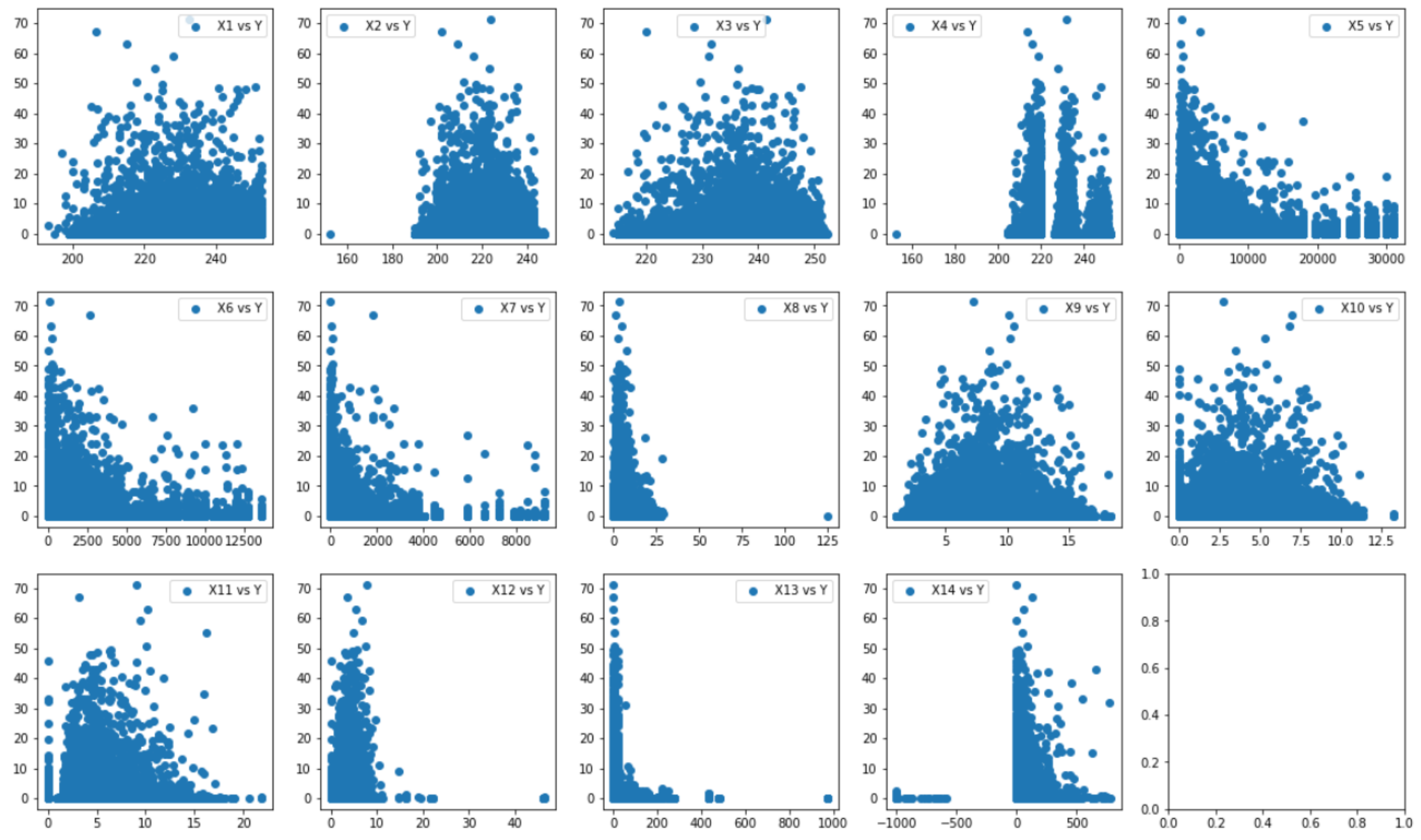
Descriptive statistics: (ρ: correlation coefficient)

Feature	Mean	Standard Deviation	Min	Max	Distribution	Correlation with Y	Strong Positive Correlation (ρ > 0.75)	Strong Negative Correlation (ρ < -0.75)
X1	242	9	193	253	Left-skewed	No apparent pattern	None	None
X2	227	10	153	248	Left-skewed	No apparent pattern	X3, X4	X9, X10
X3	242	6	214	252	Left-skewed	No apparent pattern	X2, X4	X9, X10
X4	233	10	153	252	Normally-distributed but roughly centered at three points: 220, 230, 250	Negative	X2, X3	X9, X10
X5	3,082	3,952	10	31,048	Right-skewed	Negative	X6	None
X6	921	1,738	-	13,630	Right-skewed	Negative	X5	None
X7	137	660	-	9,238	Right-skewed	Negative	None	None
X8	3	3	-	125	Right-skewed	Negative	None	None
X9	6	3	1	18	Right-skewed	No apparent pattern	X10	X2, X3, X4
X10	2	2	-	13	Right-skewed	Negative	X9	X2, X3, X4
X11	4	2	-	22	Right-skewed	Negative	X12	None
X12	3	1	-	47	Right-skewed	No apparent pattern	X11	None
X13	10	21	1	975	Right-skewed	Negative	None	None
X14	6	59	-1,000	783	Right-skewed	Negative	None	None

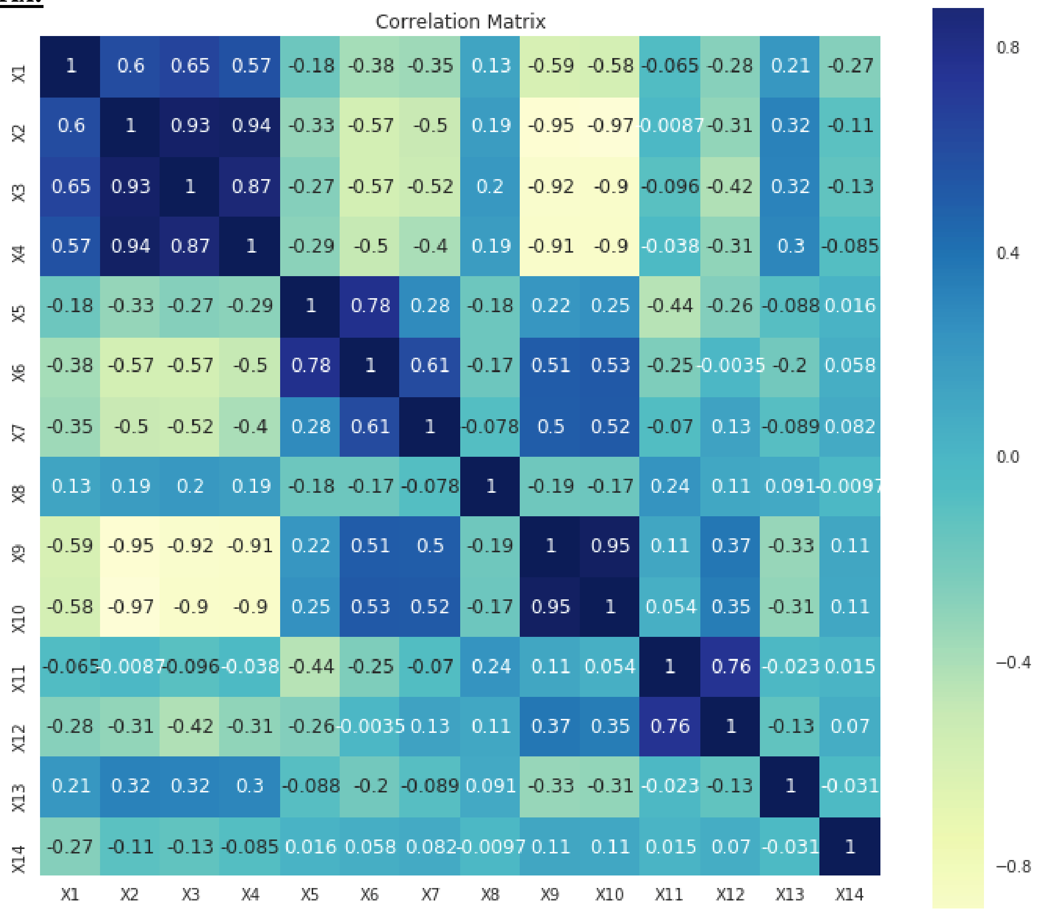
Histograms:



Scatter plots:



Correlation matrix:



	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11	X12	X13	X14
X1	1	0.6	0.65	0.57	-0.18	-0.38	-0.35	0.13	-0.59	-0.58	-0.065	-0.28	0.21	-0.27
X2	0.6	1	0.93	0.94	-0.33	-0.57	-0.5	0.19	-0.95	-0.97	-0.0087	-0.31	0.32	-0.11
X3	0.65	0.93	1	0.87	-0.27	-0.57	-0.52	0.2	-0.92	-0.9	-0.096	-0.42	0.32	-0.13
X4	0.57	0.94	0.87	1	-0.29	-0.5	-0.4	0.19	-0.91	-0.9	-0.038	-0.31	0.3	-0.085
X5	-0.18	-0.33	-0.27	-0.29	1	0.78	0.28	-0.18	0.22	0.25	-0.44	-0.26	-0.088	0.016
X6	-0.38	-0.57	-0.57	-0.5	0.78	1	0.61	-0.17	0.51	0.53	-0.25	-0.0035	-0.2	0.058
X7	-0.35	-0.5	-0.52	-0.4	0.28	0.61	1	-0.078	0.5	0.52	-0.07	0.13	-0.089	0.082
X8	0.13	0.19	0.2	0.19	-0.18	-0.17	-0.078	1	-0.19	-0.17	0.24	0.11	0.091	-0.0097
X9	-0.59	-0.95	-0.92	-0.91	0.22	0.51	0.5	-0.19	1	0.95	0.11	0.37	-0.33	0.11
X10	-0.58	-0.97	-0.9	-0.9	0.25	0.53	0.52	-0.17	0.95	1	0.054	0.35	-0.31	0.11
X11	-0.065	-0.0087	-0.096	-0.038	-0.44	-0.25	-0.07	0.24	0.11	0.054	1	0.76	-0.023	0.015
X12	-0.28	-0.31	-0.42	-0.31	-0.26	-0.0035	0.13	0.11	0.37	0.35	0.76	1	-0.13	0.07
X13	0.21	0.32	0.32	0.3	-0.088	-0.2	-0.089	0.091	-0.33	-0.31	-0.023	-0.13	1	-0.031
X14	-0.27	-0.11	-0.13	-0.085	0.016	0.058	0.082	-0.0097	0.11	0.11	0.015	0.07	-0.031	1

APPENDIX III: FORMULAS OF STATISTICS USED IN THIS PROJECT

- $Imp(X_m) = \frac{1}{N_T}\sum_T \sum_{t \in T: v(s_t)=X_m} p(t)\Delta i(s_t, t)$

  X_m = variable whose importance is evaluated
  N_T = number of trees in the forest
  T = decision tree in the forest
  t = node in the decision tree
  s_t = split of node t
  $v(s_t)$ = variable used to perform s_t
  $p(t)$ = proportion $\frac{N_t}{N}$ of samples reaching t, where N is training sample size
  $\Delta i(s_t, t)$ = impurity decrease; in sklearn, the impurity function $i(s_t, t)$ is Gini index
- $\text{VIF}_j = \frac{1}{1-{R_j}^2}$

  where ${R_j}^2$ is the coefficient of multiple determination for a regression model, using variable X_j as the dependent variable and all other X variables as independent variables
- $r^2 = \frac{Regression\ sum\ of\ squares}{Total\ sum\ of\ squares}$
- adjusted $r^2 = 1 - \left[(1-r^2)\frac{n-1}{n-k-1}\right]$

  where n is the sample size and k is the number of independent variables in the regression equation.
- $C_p = \frac{(1-{R_k}^2)(n-T)}{1-{R_T}^2} - [n - 2(k+1)]$

  k = number of independent variables included in a regression model
  T = total number of parameters (including the intercept) to be estimated in the full regression model
  ${R_k}^2$ = coefficient of multiple determination for a regression model that has k independent variables
  ${R_T}^2$ = coefficient of multiple determination for a full regression model that contains all T estimated parameters