**CS 6375**

**ASSIGNMENT: Final Project**

**Names of students in your group:**

**Rupesh Pancholi - rgp090020**

**Tyler Huning - tsh130130**

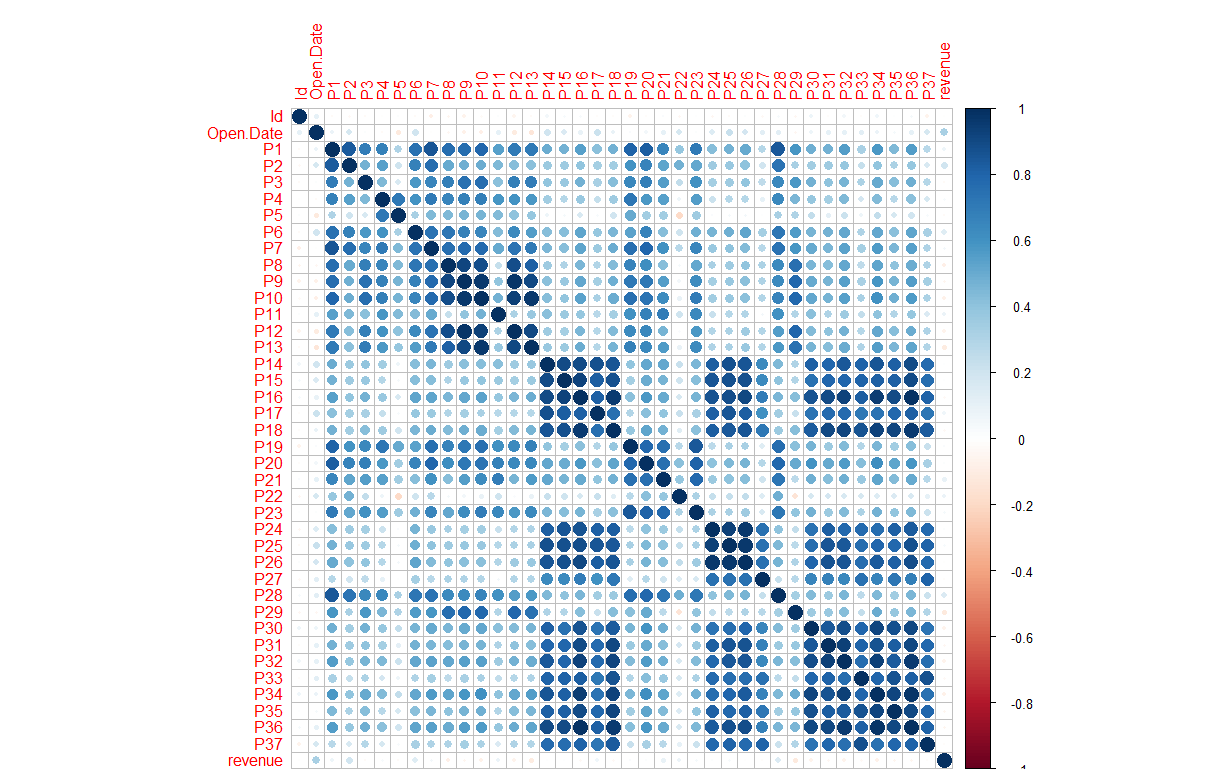
**Kruthika Vishwanath - kxv150930**

**Rakesh Balasubramani - rxb162130**

**Number of free late days used: 1**

1. **Introduction**

Restaurants can be incredibly risky investments, especially for foreign brands. Using demographic and revenue data of 137 established restaurants our objective is to predict the revenue of 100,000 restaurants in cities all over Turkey. The data was, provided by the Tab Food Investments company, from their kaggle contest Restaurant Revenue Prediction. With 20 years of history and over 1,500 restaurants it is important to learn how each city’s unique demographics factor into the success of each franchise location for the continued success of the company. Our approach was to use powerful ensemble classification techniques known for working well with instances with a large number of feature sets, along with the obfuscated demographic data to build regression models that would let us generate revenues from our quite small training set.

As with most prediction problems dealing with financial data, we needed to delve into regression models instead of the classification models that we had been practicing in class, since management in companies desire hard dollar figures as opposed to general revenue categories of simply *‘good’* or *‘bad’* investments. 



As our data is fuzzy, with no individual feature strongly correlated with the revenue, overfitting was common in our experiments, and ultimately we got a significantly lower Kaggle error by limiting the features that we used.

We found *Gradient Boosting* to give the best results, as it was able to achieve the lowest RMSE on Kaggle, and reached a rank of *379*. While SVM and Random Forest were able to achieve good rankings, Gradient Boosting was noticeably better. This can be attributed to its strengths in dealing with nonlinear separation boundaries, strong correlation between features, and its ability to deal with factor scaling.

**II. Problem Definition and Algorithm**

**II.1 *Task Definition***

The data we were provided consisted of a training dataset of 137 instances, with 43 features;

5 features define the information about the restaurant and the city that it’s in:

*Restaurant ID*, *Open Date*, *City*, *City Group* [Big Cities; Other],

*Type* [Inline, Foodcourt, Drive Through]

37 obfuscated features define the three following categories of data:

*Demographics*: Population, age, and gender distributions.

*Realestate*: Size of the restaurant (m2), front façade, and car parking availability.

*Commercial*: Proximity of points of interest such as schools, banks, and other quick service type restaurants.

1 feature defining the revenue of each operation.

Additionally we were provided with a testing dataset containing 10,000 instances with the same features as the training, bar the revenue, which we were asked to predict. Our submission was required to be in the form of a comma-separated-values file in the formatted as such:

**Restaurant ID Revenue**

0 xxxxxx

… ...

9,999 xxxxxx

This was a very interesting dataset to work, as it posed the very real and challenging problem of having, relatively, very few training instances, yet requiring high accuracy on a large number of test instances. Additionally we were required to give precise figures for the revenue, requiring regression analysis. This is the sort of problem that machine learning can be very powerful in solving, as while restaurateurs can see the correlation between a few features such as population, type of restaurant, and competition, when dealing with such a large number of seemingly unrelated features it really takes a computer to see the relationship. This problem extends well beyond restaurants, as all businesses are impacted by the same factors as these, regardless of country.

**II.2 *Algorithm Definition***

For each of the three classifiers we did preprocessing on the data. Then ran the corresponding R classifier to create a model. Then we made a prediction with the test data. That prediction was then converted to a csv file so that it can be tested on Kaggle.

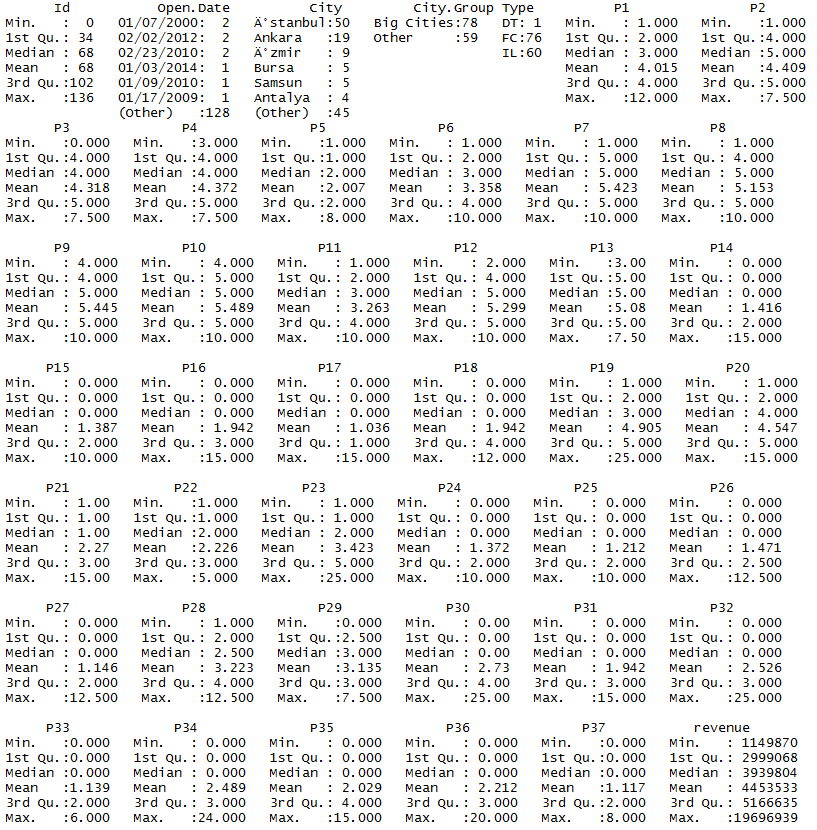
**III. Dataset description**

**Train dataset**:

Features: 42

Instances: 137

Data distribution:

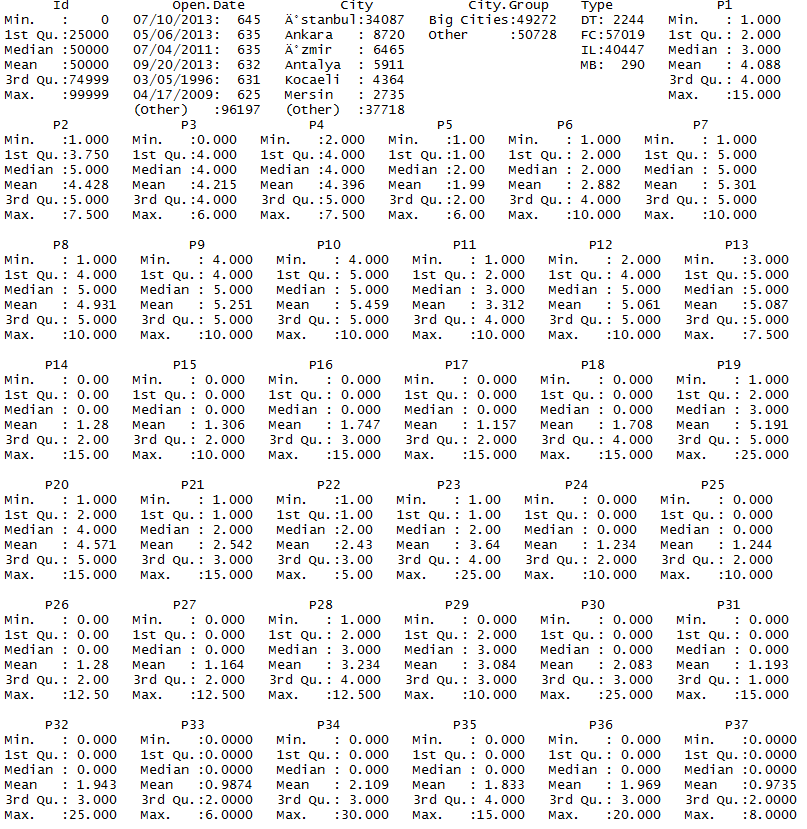


**Test dataset:**

Features: 42(no revenue provided)

Instances: 100,000

Data distribution:



**IV. Pre-processing techniques used/tried**

* Convert city group to 0/1
  + City Group had two values, “Big Cities” & “Others”. Instead of omitting this column, we changed it to 0/1 wherein Big Cities were valued at 1 and Others at 0.
* Normalize the revenue field
  + Having a threshold for revenue resulted in omitting few of the data. In order to overcome this, we planned to normalise the revenue field so that all the data would be considered.
* Changed dates to days open on May 23, 2015 (Start of the contest)
  + We found that date was a strong factor, as a restaurant open for a longer time had a proven track record of success and was able to increase customers yearly. Converting to a number allowed for us to use this factor.
* Remove: id, city, city.group, and type
  + ID was removed as there is no correlation between a restaurant’s ID and its revenue.
  + City and City group were both inconsistent between the training and testing datasets, and yielded no advantage in predicting the revenues, as only having two categories was disadvantageous to a regression type problem.
  + Type was also inconsistent between the training and testing datasets, having values in the testing set that were not in the training. Removal yielded higher results than adaptation.
* 80/20 split
  + Testing method to see how well our models performed.
* K folds
  + Testing method to see how well our models performed.
* Having individual columns for day, month, and year.
  + By splitting the dates, we were able to achieve finer detail in the strength of a restaurant’s age.
* By factoring type and city.group attributes.
  + For gradient boosting, we found the adaptation of type and city group, between training and testing datasets, to be stronger than removal.

**V. Solution, and methods**

We wanted to try three different methods to see which would work the best. The first was support vector machines. The first thing we wanted to do was get the date data into a usable format. For that we tried putting it into UTC format and days since March 23, 2015. We also tried scaling the data between 0 and 1 but it did not work as well as when we did not scale the data. This is probably do to the fact that it is a regression problem so it was more difficult to choose the max and min value when scaling the results back into a form the we could submit to kaggle. We removed data that did not seem like it influenced the outcome as much as the other data. We also tried two different splits doing a 80/20 split and a k-folds split.

Another method we used was gradient boosting machine(gbm). In this method we tried with different set of attributes and also by modifying the attributes to get better results. Firstly, we tried to remove “zero” values from the dataset by replacing it with non-zero values which helped in getting better results.

Secondly, we tried to alter the city,city.group,type attributes from the dataset, we noticed that there are few cities in the test dataset that are not present in the training dataset due to this we had to remove the city attribute before training. The date attribute was modified in several ways to get the best result, it was split up into month,year,date and number of days.

**VI. Experimental results and analysis**

**VI.1 *Results***

**1.SVM:** 80/20 split

|  |  |  |  |
| --- | --- | --- | --- |
| **Trial #** | **Kaggle Rank** | **Kaggle Error** | **Train Error** |
| 1 | 1429 | 1893585.97735 | 1862735 |
| 2 | 762 | 1843550.90450 | 2144217 |
| 3 | 1035 | 1864769.81204 | 2783807 |
| 4 | 631 | 1833517.69926 | 127914 |
| 5 | 1408 | 1890807.92789 | 3910882 |
| 6 | 1686 | 1940916.93994 | 1331602 |
| 7 | 703 | 1838937.24581 | 2524776 |
| 8 | 452 | 1822895.58306 | 1631111 |

**2. SVM:** No Split

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Trial#** | **Kernel** | **cost** | **gamma** | **Kaggle rank** | **Kaggle error** |
| 1 | radial | 1 | 1 | 1052 | 1866830.29968 |
| 2 | radial | 3 | 1 | 1661 | 1934066.38594 |
| 3 | radial | 1 | .25 | 1426 | 1893258.70105 |

**3. SVM:** 10-Fold performed

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Trial #** | **Kaggle Rank** | **Kaggle Error** | **Train Error** |  |  |
| 1 | 2213 | 3521730.12847 | 4872721 | Kernel = Radial | Scaled, outliers removed( revenue > $10 mil |
| 2 | 1456 | 1897077.10846 | 3467284 |  | Scaled, outliers kept in |
| 3 | 1427 | 1893449.39654 | 3310152 |  | Use of 1/10 test data |
| 4 | 1137 | 1874258.24518 | 3128427 | Trial 1 | 1/10 |
| 5 | 1605 | 1921139.71238 | 2428577 | Trial 2 | 2/10 |
| 6 | 1426 | 1893188.01016 | 3102187 | Trial 3 | 3/10 |
| 7 | 1620 | 1923551.29521 | 5194192 | Trial 4 | 4/10 |
| 8 | 1184 | 1878298.31571 | 3290351 | Trial 5 | 5/10 |
| 9 | 1558 | 1914498.63240 | 3290351 | Trial 6 | 6/10 |
| 10 | 1415 | 1891565.38485 | 4149498 | Trial 7 | 7/10 |

**4. Random Forest :** 10-Fold performed

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Type** | **Random Forest** | **Testing Split** | **10-Fold** | **Note** | **Note** |
| **Trial #** | **Kaggle Rank** | **Kaggle Error** | **Train Error** |  |  |
| 1 | 1460 | 1897381.73319 | 16768448 | Trial 1 | nTree = 600 |
| 2 | 1877 | 2027070.93633 | 18077311 | Trial 2 | nTree = 600 |
| 3 | 881 | 1853347.82648 | 15300288 | Trial 3 | nTree = 600 |
| 4 | 415 | 1820586.07346 | 16947463 | Trial 4 | nTree = 600 |
| 5 | 1815 | 1992998.05094 | 16947463 | Trial 5 | nTree = 600 |
| 6 | 1637 | 1928174.29421 | 16064552 | Trial 6 | nTree = 600 |
| 7 | 1822 | 1996951.39281 | 18383839 | Trial 7 | nTree = 600 |
| 8 | 1821 | 1995986.41750 | 17657810 | Trial 8 | nTree = 600 |

**5. Random Forest with no split** :

Considering the entire test (10k) data

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TRIAL** | **KAGGLE RANK** | **KAGGLE ERROR** | **TRAIN ERROR** | **NOTE** |
| 1 | 1592 | 1918775.10598 | 2571576 | Basic Model |
| 2 | 1665 | 1934999.23310 | 2668723 | ntree = 100 |
| 3 | 878 | 1852560.26011 | 2588672 | ntree=500, nodesize=17 |
| 4 | 634 | 1833610.03720 | 2595889 | ntree=1000, nodesize=25 |
| 5 | 639 | 1833970.20391 | 2568933 | ntree=10000, nodesize=25 |
| 6 | 473 | 1824543.20150 | 2590363 | ntree=5000, nodesize=50, mtry=15 |

**6. Gradient Boosting** :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **TRIAL** | **KAGGLE RANK** | **KAGGLE ERROR** | **TRAIN ERROR** | **NOTE** |
| 1 | 901 | 1855221.86776 | 1718552 | Basic model |
| 2 | 1870 | 2021638.70243 | 2260046 | Removing strings |
| 3 | 1512 | 1907890.32689 | 1587601 | Removing Outliers |
| 4 | 902 | 1855365.81877 | 2271631 | 10 Folds |
| 5 | 379 | 1817918.03889 | 2356142 | n.trees=300 |
| 6 | 414 | 1820522.58582 | 2349350 | n.trees=1 to 3000 |

**7. Gradient Boosting : 80/20 split**

|  |  |  |  |
| --- | --- | --- | --- |
| **TRIAL** | **KAGGLE RANK** | **KAGGLE ERROR** | **TRAIN ERROR WITH 80/20 SPLIT** |
| 1 | 1811 | 1991138.96570 | 2484173 |
| 2 | 571 | 1829738.49401 | 1776527 |
| 3 | 1462 | 1897743.22899 | 2564135 |
| 4 | 2001 | 2149381.32307 | 2666476 |
| 5 | 1701 | 1943913.97317 | 2338750 |

**VI.2 *Analysis***

SVM with 80/20 split did well consistently. With an average of 1866122.76123 and a lowest value of 1822895.58306. Without the split it was found that of the parameters tested the default ones with radial kernel preformed the best. SVM with kfold performed much worse than 80/20 splits with a best score of 1874258.24518 which was less than the average for the 80/20 split. On the other hand while random forest with kfold did poorly on average it was able to get a lowest value of 1820586.07346 which is quite good. Random forest with no split did quite well as well getting a lowest value of 1824543.20150 for ntree = 5000, node size = 50 and mtry = 15. Gradient boosting without 80/20 splits did quite well getting a lowest score of 1817918.03889 for ntree = 300. With 80/20 splits gradient boosting performed consistently worse.

We found Gradient Boosting to achieve the greatest accuracy due its ability to take small learning steps, and fit to the error residuals. This helps avoid the overfitting problem that our other classifiers faced, achieving good accuracy on training data set cross-validation, but failing to get very high scores in Kaggle. Gradient Boosting seems to be ideal for this type of problem, where a large number of features exist for each instance, thus the decision boundaries are quite far from linear, and features have strong correlations amongst themselves.

Additionally, due to the large number of 0 values for the features, aligned with the few number of training instances provided, dropping instances was not advisable. As gradient boosting uses decision trees as its weak classifiers, which consider each separately, we were able to scale these values out without impacting other instances. Furthermore, as we found better results including the outliers, gradient boosting meant that we experienced little effect.

**VII. Related Work**

Related problems that we encountered in our research focused on revenue or sales prediction using machine learning methods, most specifically regression type problems.

One such problem we encountered was the prediction of drug store sales based on the store brand, any current promotions, and their competition. They used autoregression along with Random Forest to create a model predicting store sales versus the features of each store on a time series model. This is similar in nature to our problem, as we too predicted sales of physical restaurant locations based on features of each location, however, our predictions are more encompassing of yearly revenue forecasts as opposed to daily. Our problem, thus, feels more practical and our solution more viable for large scale implementation for more than just restaurants. This is due to our model’s use of magnitudes more features, being sensitive to smaller changes in the retail environment.

**VIII. Conclusion**

Many models were tried and tested on this dataset and the results have been tabulated. Most of the models that was tried turned out to be unstable since the number of training instances was very less when compared to the test dataset. Learning algorithms such as SVM, Random Forest, Gradient Boosting and several others were applied on the training dataset while varying the parameters. For SVM we found that the best result were when we did a 80/20 split. It got an average score of 1866122.76123. Random Forest did best with a ntree of 5000, node size of 50 and mtry of 15. Getting a score of 1824543.20150. Gradient boosting got the best result with ntree of 300, getting a score of 1817918.03889. Overall gradient boosting was able to achieve the best results.

This could be expanded by finding a model that more accurately estimates annual restaurant revenue. It might be easier to look at a single restaurant and do forecasting based on that restaurant's specific attributes.

**IX. Contributions**

Rupesh Pancholi - Worked on the k-Fold SVM and Random Forest classifiers, also helped in writing the final report.

Tyler Huning - Worked on SVM 80/20 split and no split. Also did some preprocessing work and help with the report.

Kruthika Vishwanath - Worked on Random Forest with no split including preprocessing of the data.

Rakesh Balasubramani - Worked on Gradient boosting 80/20 split, k-folds and also no split. Used various preprocessing techniques to get the best result.

**X. References**

[1] Hinton, Kevin, and Diane Chen. "The Fundamentals of Revenue Forecasting." *The Fundamentals of Revenue Forecasting*. Pragmatic Marketing, 9 Aug. 2007. Web. 27 Nov. 2016.

[2] Mitchell, Tom M. *Machine Learning*. New York: McGraw-Hill, 1997. Print.

[3] Pei, Kevin, and Rene Bidart. "Predicting Kaggle Restaurant Annual Revenue with Support Vector Machine and Random Forest." (n.d.): n. pag. 14 Apr. 2015. Web. 27 Nov. 2016.

[4] Raul, Nataasha, Prof., Yash Shah, and Mehul Devganiya. "Restaurant Revenue Prediction Using Machine Learning." *International Journal of Engineering And Science* 6.4 (2016): n. pag. Apr. 2016. Web. 27 Sept. 2016.

[5] Xiong, Hongyu; Wu,Xi; and Yue, Jingying. “Drugs store sales forecast using Machine Learning.” http://cs229.stanford.edu/proj2015/191\_report.pdf. Web. 27 Nov. 2016.