Project Milestone	Final Project Report
Student Name	Ayush Venkatesh
	Chuyun Guan
Emails	venkatesh.ay@northeastern.edu
	guan.chu@northeastern.edu
Contribution	Ayush Venkatesh 50%
	Chuyun Guan 50%
Submission Date	August 20th, 2021

TABLE OF CONTENTS

1. Problem Setting 3

2. Problem Definition 3

3. Data Source 3

4. Data Description 3

5. Data Exploration 4

6. Data Mining Tasks 8

7. Data Mining Models/Methods/Performance. 9

8. Project Results. 16

9. Impact of the Project Outcomes. 16

10. Reference 17

Problem Setting

The garment industry is a very labor intensive industry. In addition, it requires a lot of processes and demand can get very high, especially during peak times. If a garment factory does not deliver on time, not only are customers affected, but revenue for the factory is affected as well, potentially leading to loss of jobs. In the worst case, the factory might have to close entirely. Therefore, it is important to collect data about productivity of workers and use analytics to determine important predictors of productivity.

Problem Definition

Given the above problem and dataset, where productivity is a column whose values vary between 0 and 1, the team answer the following questions:

1.) How can the team predict productivity of workers based on various features? The team will treat productivity as a continuous variable and a categorical variable, by dividing it into several classes.
2.) What predictions can the team make on future productivity based on time series?

Data Sources

In this data mining project, the data source, which is Productivity Prediction of Garment Employees Data Set, comes from the University of California - Irvine Machine Learning Repository.(1)

Data Description

The team has data from Jan 1st 2015 to March 11th 2015.

The data set contains 1197 rows and 15 columns. Following are the columns in the dataset:

(Red denotes categorical variables)

- date : Date in MM-DD-YYYY
- day : Day of the The team ek
- quarter : A portion of the month. A month was divided into four quarters
- department : Associated department with the instance (‘Sewing’ or ‘Finishing)’
- team_no : Associated team number with the instance (1-12)
- no_of_workers : Number of workers in each team
- no_of_style_change : Number of changes in the style of a particular product
- targeted_productivity : Targeted productivity set by the Authority for each team for each day. It ranges from 0-1

- smv : Standard Minute Value, it is the allocated time for a task
- wip : Work in progress. Includes the number of unfinished items for products
- over_time : Represents the amount of overtime by each team in minutes
- incentive : Represents the amount of financial incentive that enables or motivates a particular course of action.
- idle_time : The amount of time when the production was interrupted due to several reasons
- idle_men : The number of workers who were idle due to production interruption
- actual_productivity : Actual % of productivity that was delivered by the workers. It ranges from 0-1

Data Exploration

First, the team noticed that certain records in the actual_productivity column exceeded 1 despite the data source mentioning the range as 0-1. Out of 1197 records, 37 of them had this issue. In addition, the wip column had many missing values- 506 out of 1197 or 42.3%. The team ultimately removed the records where actual_productivity exceeded 1, and imputed the missing values with the median. The percentage of missing values in the wip column after dropping the productivity records exceeding 1 was only slightly higher than the percentage with those records-42.5% vs 42.3%. The team also created a column called 'month' where the values ranged from 'Jan', 'Feb' or 'March'.
All categorical columns (highlighted in red on page 3) as well as the above month column were encoded using one hot encoding, bringing the total number of columns to 33.
Let us start with the correlation heatmap. Since the number of features is very large, the team only includes the correlations between continuous predictors here. However, the team have accounted for correlations between the among variables as well as among the encoded variables and continuous variables when doing classification.

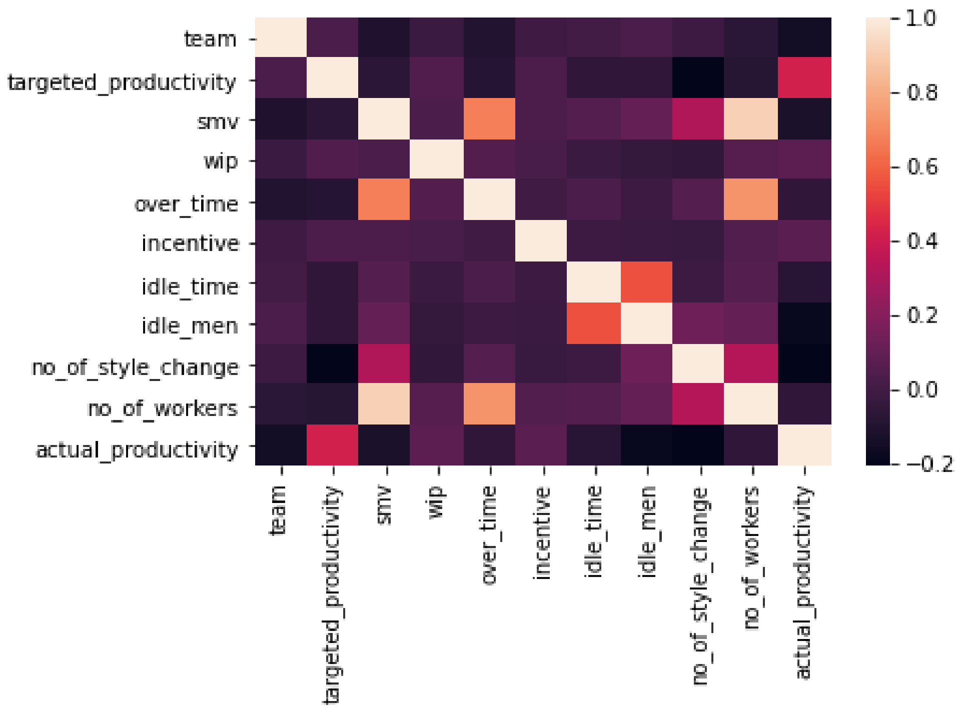


Figure 1: Correlation heatmap of variables

From the above correlation heatmap in Figure 1, the team noticed that most variables are uncorrelated, but there are some correlations present. For example, between no_of_workers and over_time. The team will come back to this correlation in a bit.

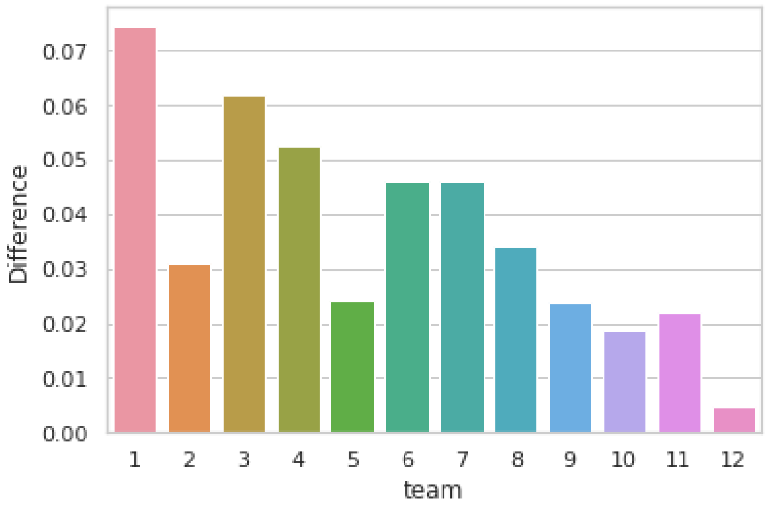


Figure 2: Bar plot of the mean difference between targeted_productivity and actual_productivity by team

From Figure 2, the team can see immediately that team 12 seems to be a real outlier. The difference between its targeted_productivity and actual_productivity is by far the smallest. What is the reason for this? Let us take a look at the actual and targeted productivity individually by team.

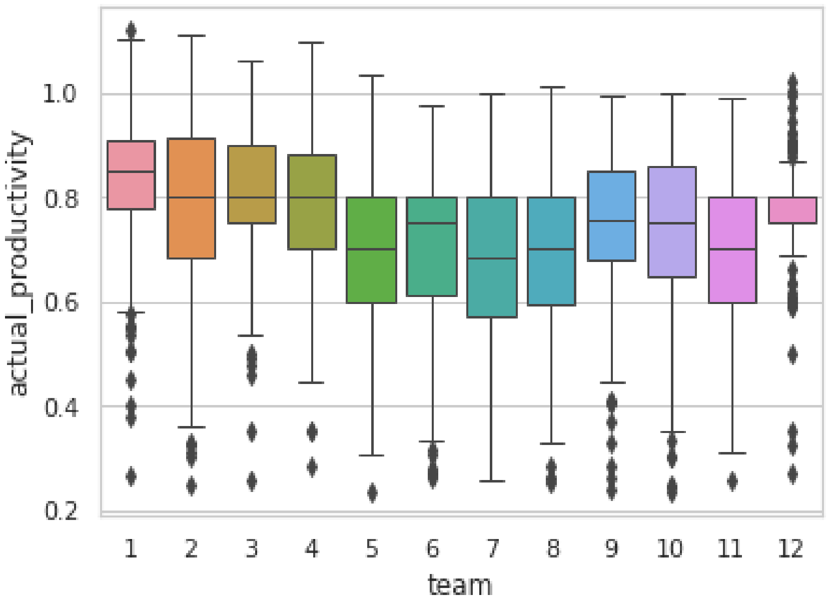


Figure 3: Boxplot of actual productivity by team

From Figure 3, although team 12 has a lot of outliers, the vast majority of the values for the actual_productivity columns are very close to each other, whereas it is quite scattered out for all the other teams.

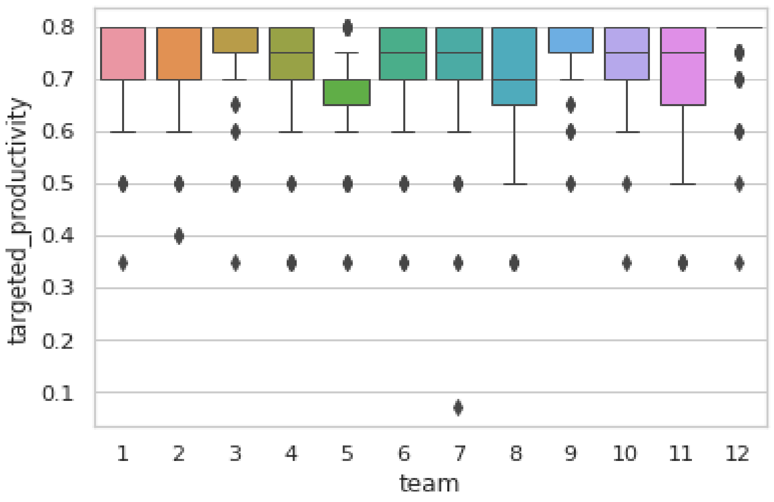


Figure 4: Boxplot for the targeted_productivity by team

Once again, in FIgure 4, the team sees that the targeted_productivity values for team 12 is far closer than any other team, and hence this explains the fact that team 12 has the lowest mean difference between actual and targeted productivity.

Coming back to the correlation heatmap, here is a scatter plot between the number of workers and overtime put in Figure 5:

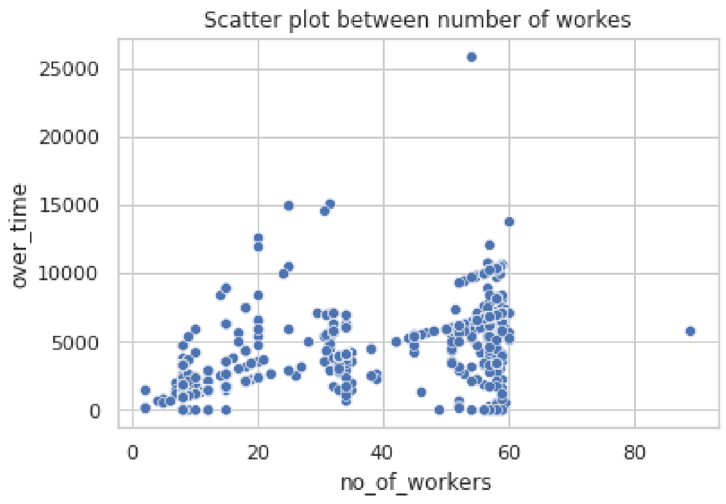


Figure 5: Scatter plot between the number of workers and overtime

As the team can see in Figure 5, as the number of workers increases, so does the overtime. However, when the team looks at the mean overtime and the mean number of workers by day, in Figure 6 and 7, an interesting trend emerges.

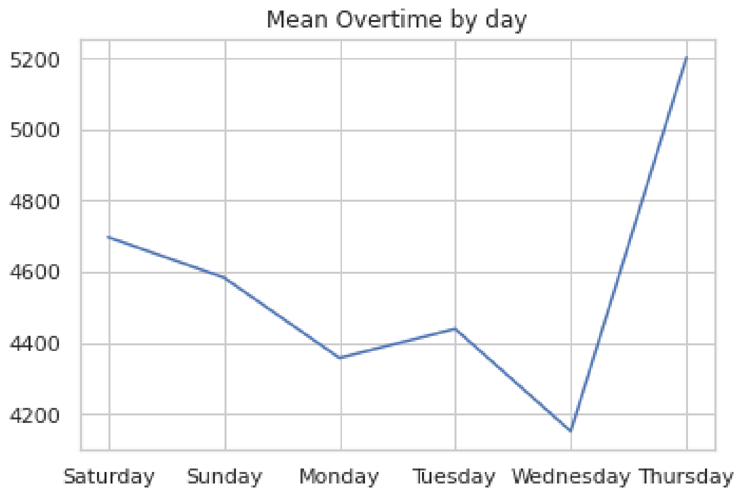


Figure 6: Line plot of the mean overtime put in per day

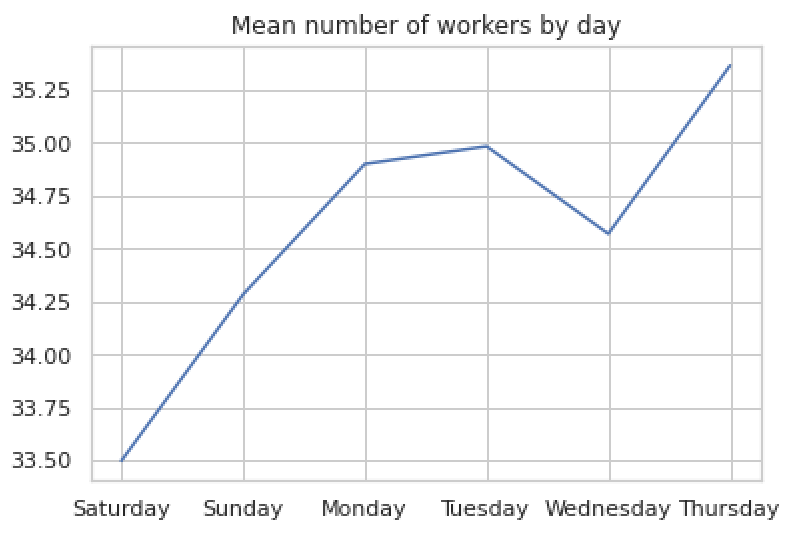


Figure 7: Mean number of workers by day

Although the mean number of workers increases per day, the overtime put in decreased, except for at the end. This trend is definitely the opposite of what the correlation heatmap suggested.

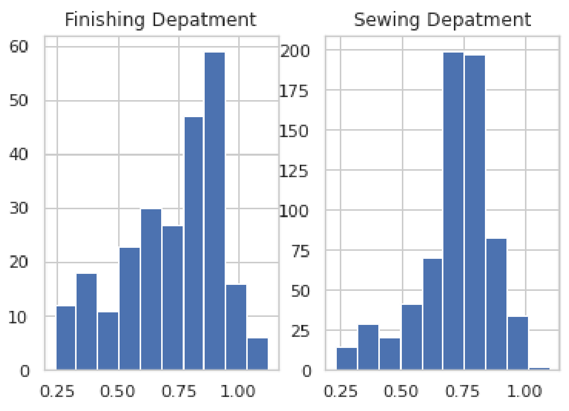


Figure 8: Histogram of the actual productivity by department

As the team can see in Figure 8, there is not a significant difference in the mean productivity between them.

Data Mining Tasks

The team will treat the actual_productivity variable in two ways:

1. As a continuous variable: The team uses Decision Trees and KNN to see if the team can predict actual_productivity.

2. As a categorical variable: The team divides into two classes which are reasonably balanced and use naive bayes and logistic regression to predict the class of productivity.

Data Mining Models/Methods/Performance

Decision Tree and KNN (using a 70/30 split for training and testing):

The first method was to try to predict actual_productivity based on the other 32 features using decision trees. The team initially did not use any hyperparameter tuning. The RMSE obtained was 0.161. Here is the histogram of the errors:

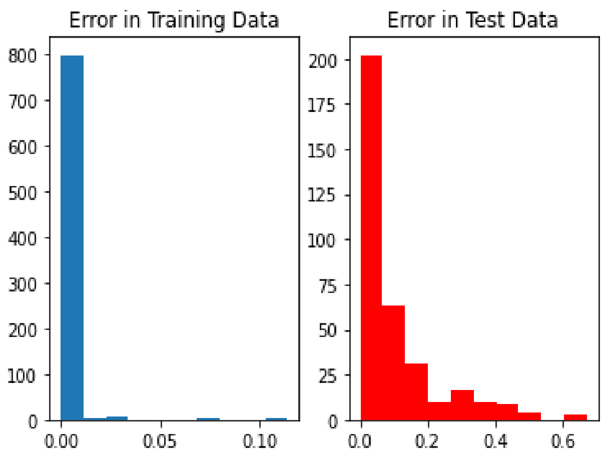


Figure 9: Histogram of Error in Training and Testing data without hyperparameter tuning

From Figure 9, the team has a very low training error, but our error in the test data is a lot higher. This suggests that overfitting occurred. So, after tuning the max_depth parameter using grid search, the RMSE obtained was considerably lower, 0.146. Here is a histogram of the errors:

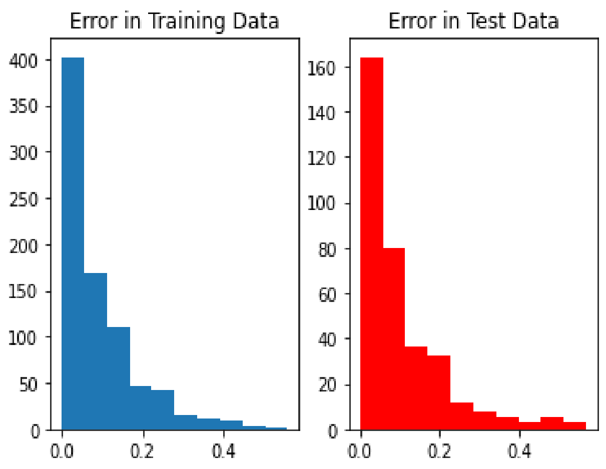


Figure 10: Histogram of Error in Training and Testing data with hyperparameter tuning

As from Figure 10, overfitting does not occur after using grid search. Now, the team has 32 features, which is a lot. Are all of them important? To see this, the team uses the feature_importance attribute. The team saw that the value for this was 0 for features 0 to 5 and features 30. So after removing these features and trying again without any hyperparameter tuning, the result was an RMSE of 0.139. Below is the histogram of the errors:

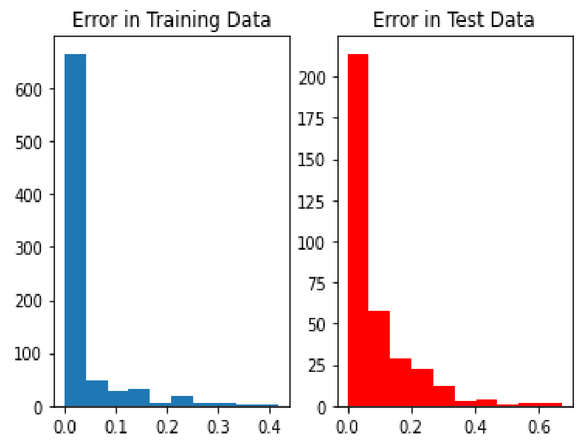


Figure 11:Histogram of Error in Training and Testing data with 32 features

Once again, the team can see that overfitting has occurred. So after tuning the hyperparameter and trying again, the team has an RMSE of 0.141, which is actually higher than the above. Here is a histogram of the errors:

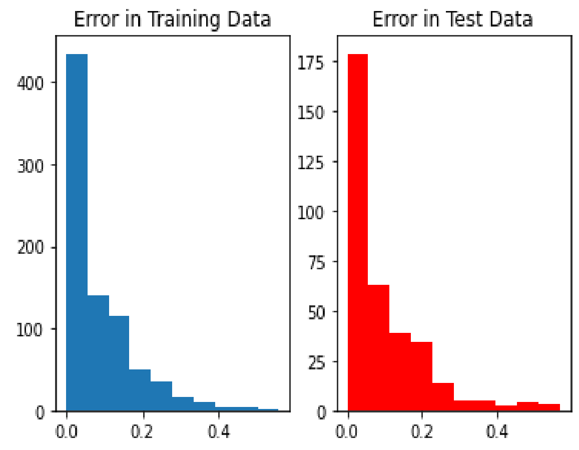


Figure 12: Histogram of Error in Training and Testing data with 32 features and hyperparameter tuning

Now, moving on to KNN, the RMSE obtained on a hyperparameter-tuned model was 0.1442. The team purposely did not try one without any hyperparameter tuning as the team has to choose a value for n_neighbours, and it is not clear what value to pick. Here is the histogram of the errors:

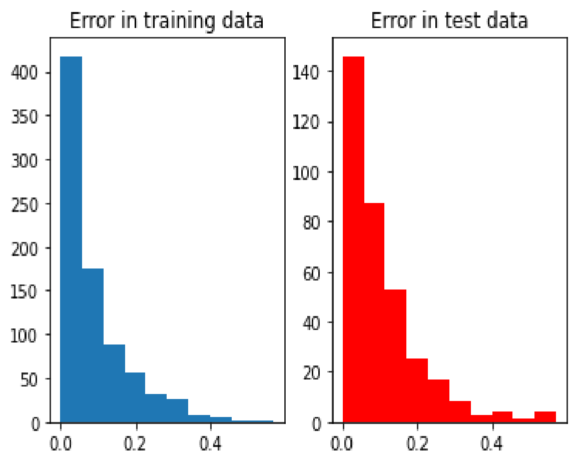


Figure 13: Histogram of Error in Training and Testing data without hyperparameter tuning

One can see that overfitting does not occur. Now, for the above, all features were used. The team would like to reduce the number of features, but there is no feature_importance score in KNN, and trying out all possibilities of predictors leads to 2^32 combinations, which is too large. So therefore, after removing the same predictors that were not important in decision trees, an RMSE of 0.1443 was obtained. This is really close to the value above, but with fewer features, showing us that these features were not important in the KNN model as well.

Logistic Regression and Naive Bayes (using a 70/30 split for training and testing):

The team created two classes from the actual_productivity column using the following rule:
if (productivity < 0.75) then 0 else 1. This creates a reasonably balanced split of 697 values for the 0 class and 463 values for the 1 class.

Starting with the Naive Bayes model. There are continuous and categorical predictors, so the team starts by using the Multinomial Naive Bayes Model. Pearsons correlation coefficient was computed between all features, and the relevant ones were removed, using a cut off of 0.7. Below is the confusion matrix using the Multinomial Model:

123	83
99	43

Table 1: Confusion matrix using the multinomial model

A low accuracy of 47% is achieved. In addition, the model does a lot better at predicting the 0 class than the 1 class. So, considering only the categorical features and using the categorical naive bayes model, here is the confusion matrix:

157	49
97	45

Table 2: Confusion matrix using the categorical naive bayes mode

This gives an accuracy of 58% which is considerably higher. Now, trying only the continuous features and using the Gaussian Naive Bayes Model, below is the confusion matrix:

196	10
131	11

Table 3: Confusion matrix using the Gaussian Naive Bayes Model with continuous features

This gives an accuracy of 59%. Moving on to Logistic Regression, and trying all features, here is the confusion matrix:

172	34
94	48

Table 4: Confusion matrix using Logistic Regression with all features

This gives an accuracy of 63%, and does much better than all the above models in the prediction of both classes. Trying only with the continuous features, the following is obtained:

192	14
118	24

Table 4: Confusion matrix using Logistic Regression with continuous features

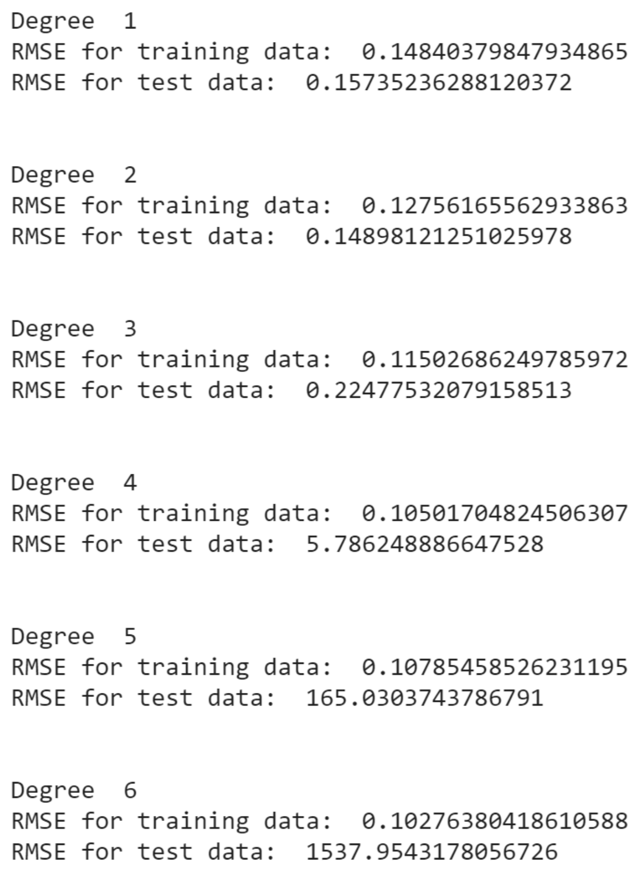
Now, this gives an accuracy of 62% with heavy misclassification of the second class. The team finally tried with only the categorical features:

165	41
103	39

Table 5: Confusion matrix using Logistic Regression with categorical features

This gives an accuracy of 58% which is much lower.

Next move to polynomial regression model:



```
Degree  1
RMSE for training data:  0.14840379847934865
RMSE for test data:  0.15735236288120372


Degree  2
RMSE for training data:  0.12756165562933863
RMSE for test data:  0.14898121251025978


Degree  3
RMSE for training data:  0.11502686249785972
RMSE for test data:  0.22477532079158513


Degree  4
RMSE for training data:  0.10501704824506307
RMSE for test data:  5.786248886647528


Degree  5
RMSE for training data:  0.10785458526231195
RMSE for test data:  165.0303743786791


Degree  6
RMSE for training data:  0.10276380418610588
RMSE for test data:  1537.9543178056726
```

Figure 14: Hyperparameter tuning: degree of polynomial regression

The polynomial regression model was conducted with only numerical data. The reason for using only numerical data and without categorical data was that categorical data does not have continuous function features while polynomial regression models require continuous function features. Figure 14 shows the RMSE results of training and testing data, from degree 1 to 6.. In conclusion, for the hyperparameter tuning of polynomial regression, degree 2 has the best performance.

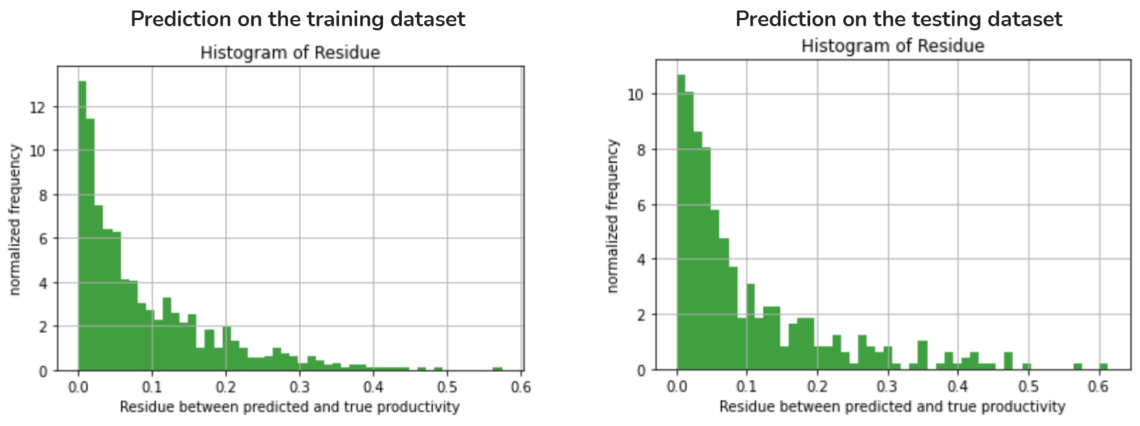


Figure 15: Visualization of the polynomial prediction on testing and training dataset

The team then conducted the visualization of the polynomial prediction, with a degree of 2. Figure X is the visualization of polynomial prediction. The left graph is a visualization of prediction on the training dataset, while the right graph on the testing dataset. The X-axis is the frequency and the y-axis is the residue. As a result, the polynomial regression model of testing and training dataset performs well.

Time Series for predictions:

The team conducted time series method for prediction of productivity. With time series, the team performed two predictions.

Prediction 1:

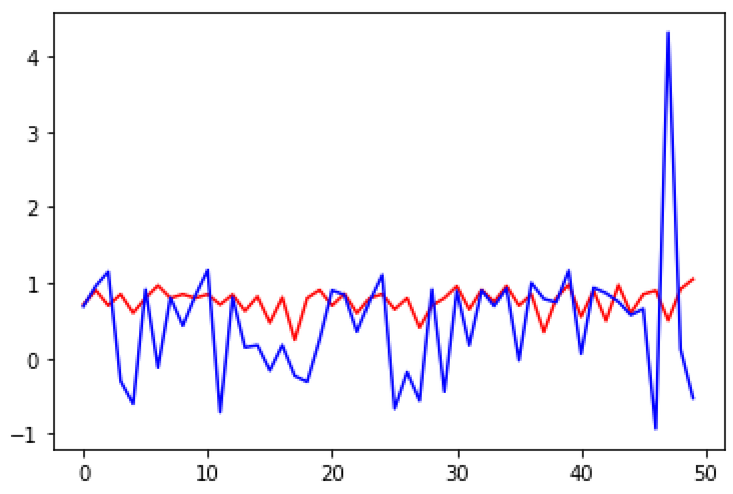


Figure 16: Line chart of average prediction

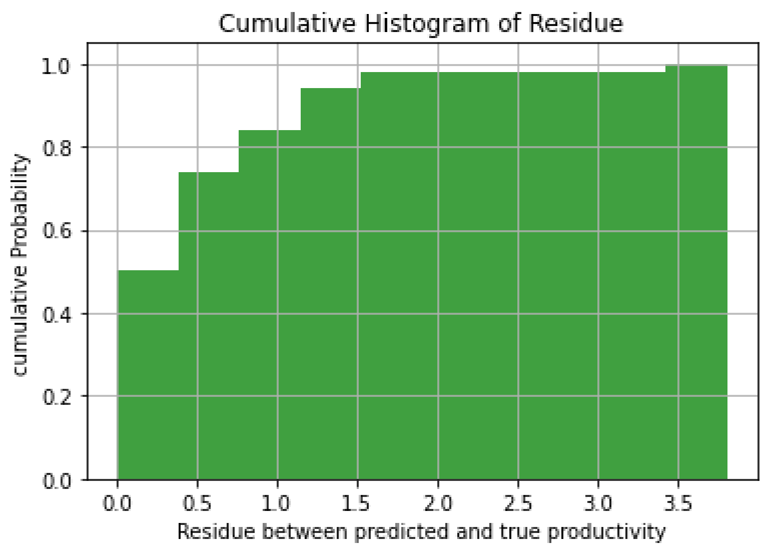
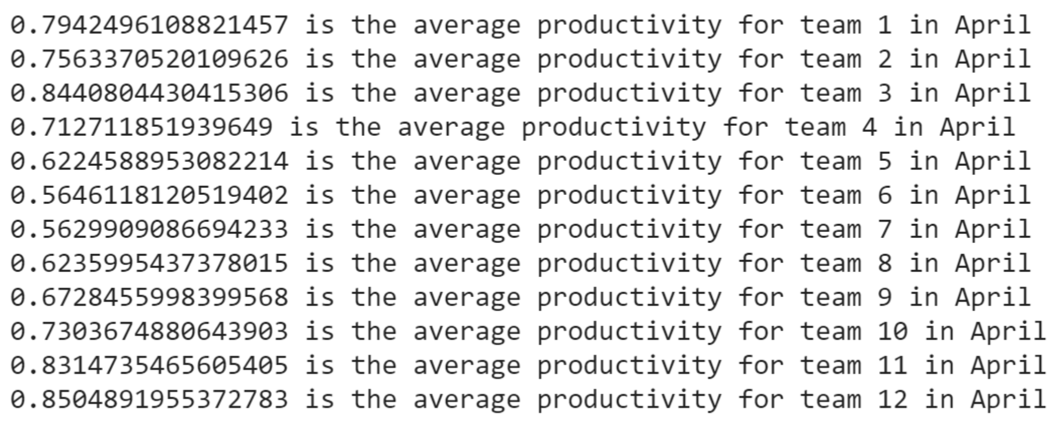


Figure 17: Cumulative Histogram of Residue

The team has used the last 11 teams to predict the next team based on the time series data 50 times. For example, the productivity prediction of team 12 is based on the data of the productivity prediction of team 1 to 11. Then plotted the average prediction on Figure 16. The blue line is the predicted productivity and red is the ground true. As shown, the trend is almost the same. However, the last prediction is entirely wrong and that may be an outlier in the model. Figure 17 is the cumulative histogram of residue for this prediction, and the performance is great.

Prediction 2:



```
0.7942496108821457 is the average productivity for team 1 in April
0.7563370520109626 is the average productivity for team 2 in April
0.8440804430415306 is the average productivity for team 3 in April
0.712711851939649 is the average productivity for team 4 in April
0.6224588953082214 is the average productivity for team 5 in April
0.5646118120519402 is the average productivity for team 6 in April
0.5629909086694233 is the average productivity for team 7 in April
0.6235995437378015 is the average productivity for team 8 in April
0.6728455998399568 is the average productivity for team 9 in April
0.7303674880643903 is the average productivity for team 10 in April
0.8314735465605405 is the average productivity for team 11 in April
0.8504891955372783 is the average productivity for team 12 in April
```

Figure 18: Productivity prediction of teams on April

The second prediction of time series used the productivity data of each team, from January to March, to predict the productivity in April for each team. The productivity predictions for April for each time are shown in Figure 18. And from the data, one can conclude that team 12 has the best performance, while team 7 has the worst.

Project Results

In conclusion for the data mining project, the best performing model when productivity is treated as a continuous variable is the unoptimized decision tree with irrelevant features removed, because it has the lowest RMSE, which is 0.139. On the other side, the best performing model, when productivity is treated as a categorical variable, is the logistic regression with categorical and continuous features, because it has the highest accuracy, which is 63%.

Impact of the Project Outcomes

The value created by the models was that the team got to know what the most important features were. By eliminating them, the team created a much simpler model to predict productivity. In addition to directly predicting productivity, this was turned into a classification problem, where the productivity column was divided into two categories, making for a simpler analysis. There were some challenges that the team faced in this project. First, there are lots of missing values, since different ways of imputing, modelling will give different results. Second, there are lots of categorical variables, so after encoding, the number of features is large, totally 33 of them with a relatively small number of records. Lastly, as a result of the small number of records, the training and testing set is also small.

Reference

1. Imran, Abdullah Al. “Productivity Prediction of Garment Employees Data Set.” *UCI Machine Learning Repository*, 3 Aug. 2020, archive.ics.uci.edu/ml/datasets/Productivity+Prediction+of+Garment+Employees.