Predicting the popularity of merchandise by creating the machine learning classification model

Siddharth Paraag Nilakhe
Dept. of Data Science
Stevens Institute of Technology
Hoboken, US
snilakhe@stevens.edu

Mehul Sudhakar Ingale
Dept. of Data Science
Stevens Institute of Technology
Hoboken, US
mingale1@stevens.edu

Preeti Reddy Koppolu
Dept. of Data Science
Stevens Institute of Technology
Hoboken, US
pkoppolu@stevens.edu

***Abstract*—Merchandise business has grown exponentially during the last decade, and the industries are focusing on the merchandise business more than before. However, just setting up an online store and starting selling might not work. Different machine learning techniques are needed to know the users' preferences and know what would be best for business. The popularity class decides how popular the product is given the attributes which a store owner can control. Data gathered from one of the top apparel brands in India has been put through its paces. In this project we developed a model that predicts the popularity of merchandise by a given dataset. The popularity is based on several factors like Store ratio, category 1, category 2, store score, basket score and predicts the most important feature that will determine the success of popularity.**

I. Introduction

Big Brands spend a significant amount on popularizing a product. Nevertheless, their efforts go in vain while establishing the merchandise in the hyperlocal market. Based on different geographical conditions same attributes can communicate a piece of much different information about the customer. Hence, insights this is a must for any brand owner. We will be solving this problem by using various machine learning algorithms.

II. Related Work

The information was taken from a closed contest at https://machinehack.com/hackathons/merchandise_popularity_prediction_challenge/data. The number of participants in the competition was roughly 680. To get the most out of the data, we will compare the conclusions we have reached to those of the competition winners.

III. Our Solution

A. Description of Dataset

The dataset has been obtained from https://machinehack.com/hackathons/merchandise_popularity_prediction_challenge/data. The dataset consists of 15 features and 12228 records. The author of the dataset has not provided a clear description of the attributes. The data description can be considered anonymized. However, these attributes are limited to the business use case and there are no other variables that are collected and can Impact the Popularity of the merchandise. The following description is based on our analysis of the dataset.

- Basket-Ratio - Number of items getting sold in a single purchase.
- Score-1, score-2, score-3, score-4 - Four types of scores allocated to the merchandise that affect the popularity directly or indirectly.
- Store-score - Score given to a particular store based on several factors.
- Store-presence - Presence of the store in the market with respect to others.
- Time - Time spent in popularizing the merchandise.
- Popularity - Target variable

B. Pre-processing techniques:

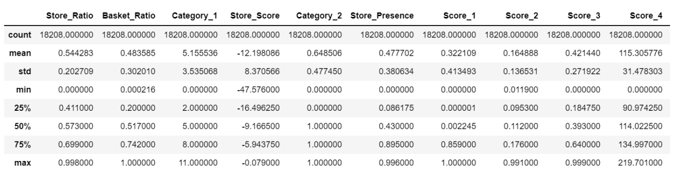
We have defined a function called clean data, which is used for preprocessing the data. This function takes in a dataframe as an input and performs several operations on it including:

- Removing any rows with null values
- Adding a new column called time of day which is derived from the time column
- Encoding the Category 1, Category2, and time of day columns using one hot encoding
- Scaling the Store Ratio, Basket Ratio, and Store Score columns using min max scaling
- Then we call the clean data function on both the train and test dataframes and assign the returned values to the variables X train, X test, y train, and y test.
- Duplicates were eliminated from the dataset.
- Changed the unit of time from seconds to years, days, hours, and minutes.
- Scaled the features ”Store Ratio,” ”Basket Ratio,” ”Store Score,” ”Store Presence,” ”Score 1,” ”Score 2,” ”Score 3,” and ”Score 4”.

Exploratory data analysis:

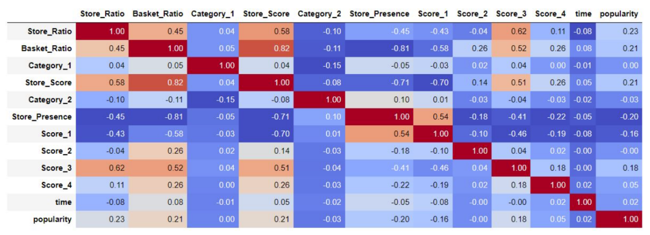
This table show mathematical relations of the train dataset like count, min, max values,std, etc.

- Store ratio is strongly correlated with basket ratio, store score, store presence, score1, score3.



	Store_Ratio	Basket_Ratio	Category_1	Store_Score	Category_2	Store_Presence	Score_1	Score_2	Score_3	Score_4
count	18208.000000	18208.000000	18208.000000	18208.000000	18208.000000	18208.000000	18208.000000	18208.000000	18208.000000	18208.000000
mean	0.544283	0.483585	5.155536	-12.198086	0.648506	0.477702	0.322109	0.164888	0.421440	115.305776
std	0.202709	0.302010	3.535068	8.370566	0.477450	0.380634	0.413493	0.136531	0.271922	31.478303
min	0.000000	0.000216	0.000000	-47.576000	0.000000	0.000000	0.000000	0.011900	0.000000	0.000000
25%	0.411000	0.200000	2.000000	-16.496250	0.000000	0.086175	0.000001	0.095300	0.184750	90.974250
50%	0.573000	0.517000	5.000000	-9.166500	1.000000	0.430000	0.002245	0.112000	0.393000	114.022500
75%	0.699000	0.742000	8.000000	-5.943750	1.000000	0.895000	0.859000	0.176000	0.640000	134.997000
max	0.998000	1.000000	11.000000	-0.079000	1.000000	0.996000	1.000000	0.991000	0.999000	219.701000

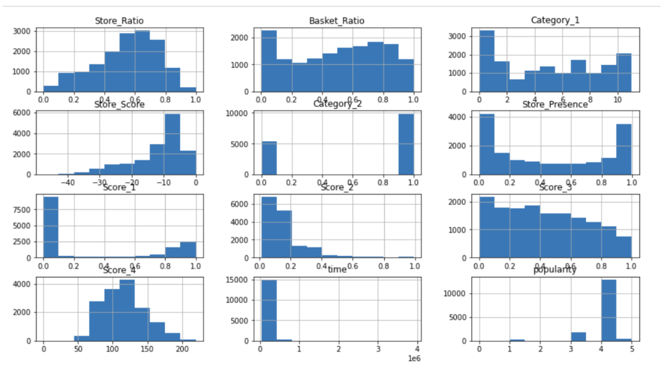
Fig. 1. Relations of Train dataset



	Store_Ratio	Basket_Ratio	Category_1	Store_Score	Category_2	Store_Presence	Score_1	Score_2	Score_3	Score_4	time	popularity
Store_Ratio	1.00	0.45	0.04	0.58	-0.10	-0.45	-0.43	-0.04	0.62	0.11	-0.08	0.23
Basket_Ratio	0.45	1.00	0.05	0.82	-0.11	-0.81	-0.58	0.26	0.52	0.26	0.08	0.21
Category_1	0.04	0.05	1.00	0.04	-0.15	-0.05	-0.03	0.02	0.04	0.00	-0.01	0.00
Store_Score	0.58	0.82	0.04	1.00	-0.08	-0.71	-0.70	0.14	0.51	0.26	0.05	0.21
Category_2	-0.10	-0.11	-0.15	-0.08	1.00	0.10	0.01	-0.03	-0.04	-0.03	-0.02	-0.03
Store_Presence	-0.45	-0.81	-0.05	-0.71	0.10	1.00	0.54	-0.18	-0.41	-0.22	-0.05	-0.20
Score_1	-0.43	-0.58	-0.03	-0.70	0.01	0.54	1.00	-0.10	-0.46	-0.19	-0.08	-0.16
Score_2	-0.04	0.26	0.02	0.14	-0.03	-0.18	-0.10	1.00	0.04	0.02	-0.00	-0.00
Score_3	0.62	0.52	0.04	0.51	-0.04	-0.41	-0.46	0.04	1.00	0.18	-0.00	0.18
Score_4	0.11	0.26	0.00	0.26	-0.03	-0.22	-0.19	0.02	0.18	1.00	0.02	0.05
time	-0.08	0.08	-0.01	0.05	-0.02	-0.05	-0.08	-0.00	-0.00	0.02	1.00	0.02
popularity	0.23	0.21	0.00	0.21	-0.03	-0.20	-0.16	-0.00	0.18	0.05	0.02	1.00

Fig. 2. Correlation Matrix

- Basket ratio is correlated to store score, store presence, score1, score3.
- Score presence is correlated with score1, score2, score3, popularity



captionHistogram

- The data distribution of store presence is bimodal.
- Score_2, score 1, score3 and time data distribution is skewed left.
- Store score and popularity data distribution is skewed right.
- Distribution of basket ratio is multimodal.
- Distribution of store ratio is bimodal and skewed.

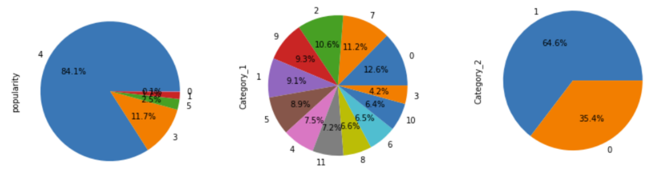


Fig. 3. Pie Chart

Popularity score 4 has been assigned to 84.1% of the merchadise. There is a increase in store score when there is a

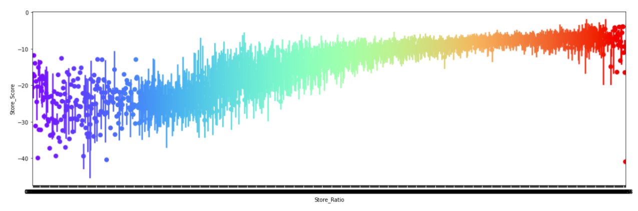


Fig. 4. Point plot between store score and store ratio

increase in store ratio There is a increase in basket ratio when

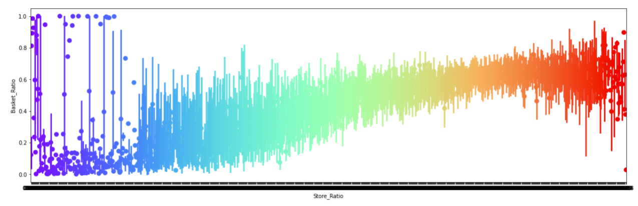


Fig. 5. Point plot between basket ratio and store ratio

there is a increase in store ratio. There is a increase in store

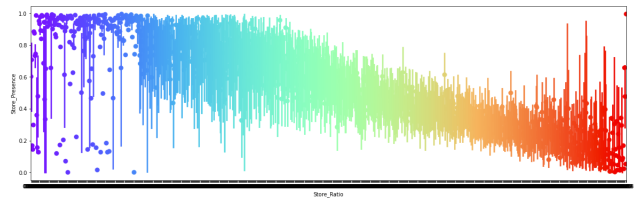


Fig. 6. Point plot between store ratio and store presence

ratio when there is a decrease in store presence. There is a

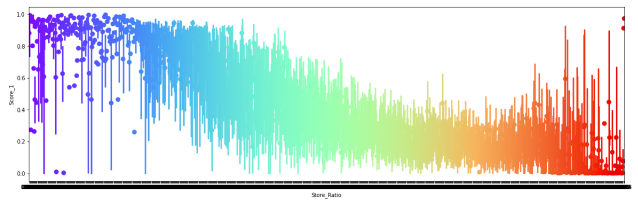


Fig. 7. Point plot between score and store ratio

decrease in score_1 when there is a increase in store ratio.

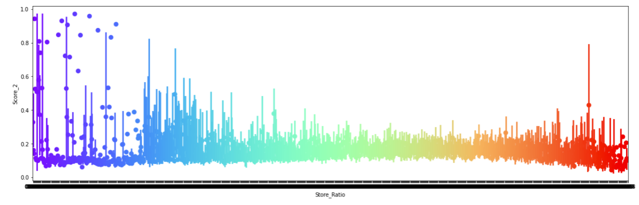


Fig. 8. Point plot between Store Ratio and Score 2

C. Machine Learning Algorithms

1) Logistic Regression: Logistic regression, the most fundamental approach for classification, will be used as our first step.

Predictive analytics and categorization frequently make use of this kind of statistical model, also referred to as a logit model. Based on a given dataset of independent variables, logistic regression calculates the likelihood that an event will occur, such as voting or not voting. Given that the result is a probability, the dependent variable's range is 0 to 1. In logistic regression, the odds—that is, the probability of success divided by the probability of failure—are transformed using the logit formula.

Why have we chosen Logistic Regression as our first model?

There are several reasons why we have chosen logistic regression as our first approach to the given problem:

- It is a simple and easy to implement algorithm: Logistic regression is based on a straightforward linear model, so it is relatively easy to understand and implement.
- It can handle binary and multi class classification: Logistic regression can be used for both binary and multi class classification tasks.
- It is fast and efficient: Logistic regression is relatively fast to train and make predictions with, making it suitable for large datasets.
- It can perform well with a small number of samples: Logistic regression can still perform well even with a small number of samples, as long as the samples are representative of the population.
- It provides interpretable results: Logistic regression provides coefficients for each feature, which can be used to understand how each feature impacts the outcome.
- Overall, logistic regression is a reliable and effective algorithm for classification tasks and is a good choice for many applications.

2) Random Forest Classifier: Random Forest Classifier is a popular ensemble learning method for classification tasks. It works by training multiple decision trees on random subsets of the training data and then aggregating the predictions of the individual decision trees to make a final prediction. This approach can help to reduce overfitting, improve generalization to new data, and increase the model's performance.

We fit the Random Forest Classifier model to the training data using the RandomForestClassifier class from the sklearn.ensemble module. The model is instantiated with the specified number of estimators (200) and maximum depth (40) for the decision trees. The model is then fit to the training data using the fit method.

The print score function is then called to print the performance scores of the model on the training and test datasets. The visualize importance function is used to visualize the feature importances, which represent the relative importance of each feature in the model. The feature importances are obtained using the feature importances attribute of the model.

We use Random Forest classifier as our second approach to solve the given problem because of the following reasons:

- Improved performance: Random Forest classifiers are known to have good generalization performance and can often achieve higher accuracy compared to single decision trees. This is because the ensemble approach of aggregating the predictions of multiple decision trees can reduce overfitting and improve the robustness of the model.
- Interpretability: Although Random Forest classifiers are generally less interpretable than single decision trees, they can still provide some insight into the relative importance of different features in the model. The feature importances can be obtained using the feature importances attribute of the model, which can be useful for understanding the factors that influence the model's predictions.
- Fast training and prediction times: Random Forest classifiers are relatively fast to train and make predictions with, even on large datasets. This makes them a good choice for applications where speed is a concern.
- Ability to handle large datasets: Random Forest classifiers are able to handle large datasets effectively, as the individual decision trees can be trained in parallel. This can make them a good choice for applications with very large datasets.
- Ability to handle high-dimensional data: Random Forest classifiers are able to handle high-dimensional data effectively, as the decision trees can be constructed based on a subset of the features at each split. This makes them a good choice for applications with many features.

3) K-Nearest Neighbors: KNN (K-Nearest Neighbors) is a supervised machine learning algorithm that is used for both classification and regression. It works by finding the K-nearest data points (number of data points is represented by the variable K) to a given data point, and then classifying that data point based on the class of the majority of the K-nearest data points.

In KNN classification, the data is divided into training and test datasets, and the model is trained using the training dataset. During the testing phase, the model predicts the class of a given data point based on the classes of its K-nearest neighbors in the training dataset. The model's prediction is made based on the majority class among the K-nearest neighbors.

KNN is a simple and effective algorithm that is easy to implement and can be used for various classification tasks. However, it can be computationally expensive, as the model needs to calculate the distance between the test data point and all the data points in the training dataset in order to find the K-nearest neighbors.

It is also sensitive to the scale of the features, and requires normalization of the data in order to work effectively.

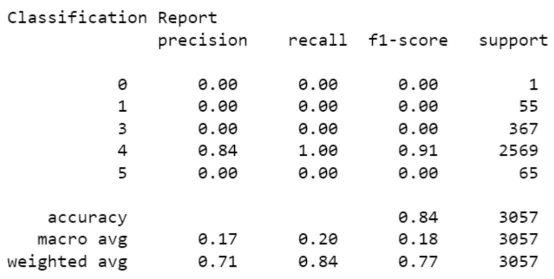
4) Voting Classifier: Voting Classifier is an ensemble method that combines the predictions from multiple machine learning models and returns the prediction with the most votes. It can be used for both classification and regression tasks. It is a high bias low variance classifier.

In the code, we have defined a list of classifier objects called 'classifiers' which consists of knn clf, svc, dt clf and rf clf. These are the models that will be used for voting. Then, the Voting Classifier model is instantiated and the classifiers list is passed to the 'estimators' parameter. The 'voting' parameter is set to 'hard', which means that the final prediction will be based on the mode of the class labels predicted by the individual classifiers. Finally, the Voting Classifier model is fit to the training data and the performance of the model is printed using the 'print score' function.

D. Implementation Details

1) Logistic Regression : As mentioned in the above section, we used logistic regression as our first approach towards solving the problem. The classification report provides performance metrics for a classification model on a test dataset. The metrics include precision, recall, f1-score, and support for each class.

In classification report given by logistic regression we can clearly see that there are five classes: 0, 1, 3, 4, and 5. The model appears to have performed well for class 4, with a precision of 0.84, recall of 1.00, and f1-score of 0.91. However, the model did not perform well for the other classes, with precision, recall, and f1-scores all equal to 0.0.



```
Classification Report
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        55
           3       0.00      0.00      0.00       367
           4       0.84      1.00      0.91      2569
           5       0.00      0.00      0.00        65

    accuracy                           0.84      3057
   macro avg       0.17      0.20      0.18      3057
weighted avg       0.71      0.84      0.77      3057
```

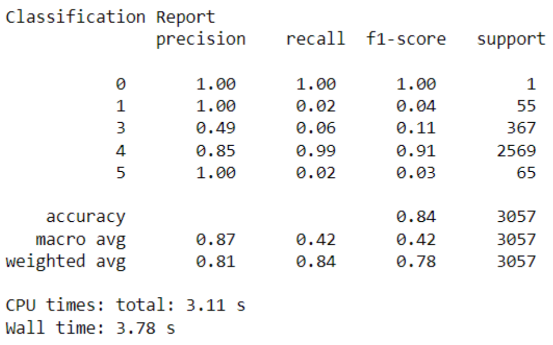
Fig. 9. Classification report of Logistic Regression

The overall accuracy of the model is 0.84, which means that the model correctly predicted the class for 84% of the test samples. The macro average of the precision, recall, and f1-scores is 0.18, which is lower than the accuracy, indicating that the model is not performing equally well for all classes. The weighted average of the precision, recall, and f1-scores is 0.77, which is also lower than the accuracy.

The support for each class indicates the number of samples in the test set that belong to each class. Class 4 has the highest support with 2569 samples, followed by class 3 with 367 samples, class 5 with 65 samples, class 1 with 55 samples, and class 0 with 1 sample.

Overall, the classification report suggests that the model is performing poorly for classes 0, 1, 3, and 5, and well for class 4. Improving the performance of the model for the poorly performing classes may be necessary in order to achieve better overall performance

2) Random Forest Classifier: In comparison to the previous classification report that we received for Logistic Regression, this report shows improved performance for class 0, with a precision, recall, and f1-score of 1.0. However, the performance for class 1 has decreased, with a precision of 1.0 but a low recall of 0.02 and f1 score of 0.04. The performance for class 3 has also decreased, with a precision of 0.57, recall of 0.06, and f1-score of 0.11. Class 4 continues to show good performance, with a precision of 0.85, recall of 0.99, and f1-score of 0.92. Class 5 has a precision of 1.0 but a low recall of 0.02 and f1 score of 0.03.



```
Classification Report
              precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      0.02      0.04        55
           3       0.49      0.06      0.11       367
           4       0.85      0.99      0.91      2569
           5       1.00      0.02      0.03        65

    accuracy                           0.84      3057
   macro avg       0.87      0.42      0.42      3057
weighted avg       0.81      0.84      0.78      3057

CPU times: total: 3.11 s
Wall time: 3.78 s
```

Fig. 10. Classification report of Random Forest Classifier

Overall, the model has improved performance for class 0 but has decreased performance for class 1 and 3 compared to the previous report. The performance for class 4 remains good, while the performance for class 5 is still poor. The accuracy of the model is unchanged at 0.84, but the macro average of the precision, recall, and f1 scores has improved to 0.42, and the weighted average has decreased slightly to 0.78.

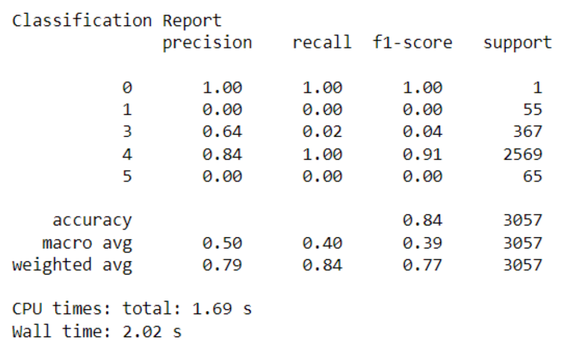
In summary, the model has improved performance for some classes but has decreased performance for other classes compared to the previous report. Further improvement may be necessary in order to achieve better overall performance for the model.

3) Random Forest Classifier using Randomized Search CV: Randomized Search CV is a technique for hyperparameter optimization in which random combinations of the hyperparameters are used to train the model and the best combination is chosen based on the performance of the model. The purpose of this process is to find the best combination of hyperparameters that can lead to optimal performance of the model. We used Randomized Search CV to find out the best hyperparameters for Random Forest Classifier

In the first classification report(without Randomized Search CV), the precision, recall and f1-score for class 1 are higher compared to the second classification report(with Randomized Search CV) . However, the precision, recall and f1-score for class 3 and class 5 are lower in the first classification report compared to the second classification report. Overall, the accuracy is the same for both classification reports, but the macro average and weighted average scores are slightly higher in the second classification report.

Hence using Randomized Search CV did not have any great impact on the results as we expected.

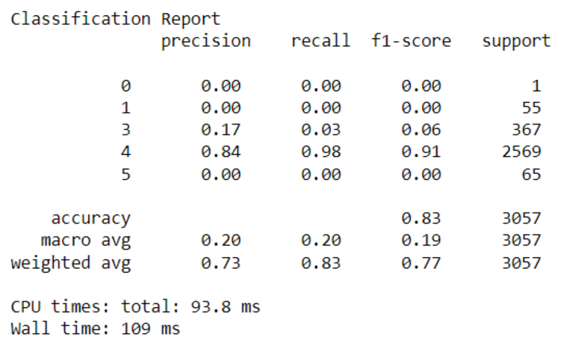
4) K-Nearest Neighbors : In comparing the classification report for Random Forest Classifier to the classification report for KNN, we can see that the Random Forest Classifier generally performs better in terms of precision, recall, and f1-score for all classes. The accuracy of the Random Forest



```
Classification Report
                   precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       0.00      0.00      0.00        55
           3       0.64      0.02      0.04       367
           4       0.84      1.00      0.91      2569
           5       0.00      0.00      0.00        65

    accuracy                           0.84      3057
   macro avg       0.50      0.40      0.39      3057
weighted avg       0.79      0.84      0.77      3057

CPU times: total: 1.69 s
Wall time: 2.02 s
```

Fig. 11. Classification report of Random Forest Classifier using RandomCV

Classifier is also slightly higher than that of the KNN classifier



```
Classification Report
                   precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        55
           3       0.17      0.03      0.06       367
           4       0.84      0.98      0.91      2569
           5       0.00      0.00      0.00        65

    accuracy                           0.83      3057
   macro avg       0.20      0.20      0.19      3057
weighted avg       0.73      0.83      0.77      3057

CPU times: total: 93.8 ms
Wall time: 109 ms
```

Fig. 12. Classification report of K-Nearest Neighbor

However, it is important to note that the performance of a machine learning model depends on various factors such as the nature of the data, the quality of the features, and the choice of hyperparameters, among others.

t is possible that the KNN classifier may perform better on a different dataset or with different hyperparameter settings.

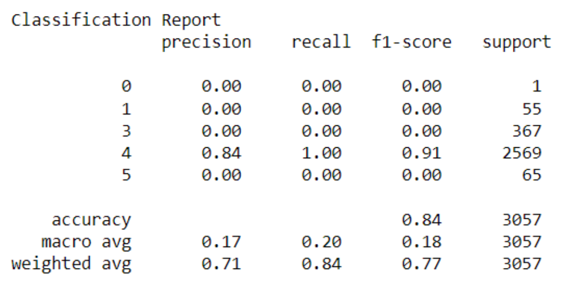
It is always a good idea to try out multiple models and compare their performance to choose the best one for a particular problem.

5) K-Nearest Neighbors using Randomized Search CV: We try and use Randomized Search CV on KNN to find out the best combination of hyperparameters to improve our accuracy.

The Classification Report for KNN after applying Randomized Search CV shows a slightly better performance compared to the original KNN model. The precision for class 4 has improved from 0.84 to 1.0, and the recall for class 4 has also improved from 0.98 to 1.0. The f1-score for class 4 has also improved from 0.91 to 0.91.

However, the performance for the other classes has not improved and remains the same or worse compared to the original KNN model. Overall, the accuracy of the model has improved slightly from 0.83 to 0.84.

However, it is worth noting that the improvement in performance may not be significant enough to justify the additional computational cost of applying Randomized Search CV. It is also possible that the original KNN model may have already



```
Classification Report
                   precision    recall  f1-score   support

           0       0.00      0.00      0.00         1
           1       0.00      0.00      0.00        55
           3       0.00      0.00      0.00       367
           4       0.84      1.00      0.91      2569
           5       0.00      0.00      0.00        65

    accuracy                           0.84      3057
   macro avg       0.17      0.20      0.18      3057
weighted avg       0.71      0.84      0.77      3057
```

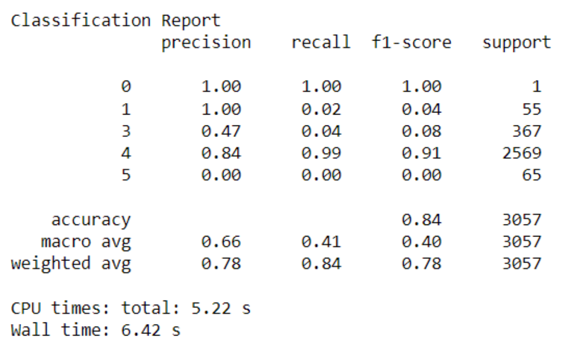
Fig. 13. Classification report of K-Nearest Neighbor using RandomCV

been performing adequately for the given data, and applying Randomized Search CV may not have resulted in a significant improvement in performance.

It is always important to carefully evaluate the trade-off between the computational cost and the potential improvement in model performance when deciding whether or not to apply techniques such as hyperparameter optimization.

6) Voting Classifier: The classification report above shows the performance of a voting classifier on a dataset with five classes: 0, 1, 3, 4, and 5. The classifier has made predictions for each of these classes, and the report shows the precision, recall, f1-score, and support for each class.

Precision is a measure of the classifier's ability to accurately predict the class. It is the number of true positive predictions (i.e., predictions that were correct) divided by the total number of positive predictions made by the classifier.



```
Classification Report
                   precision    recall  f1-score   support

           0       1.00      1.00      1.00         1
           1       1.00      0.02      0.04        55
           3       0.47      0.04      0.08       367
           4       0.84      0.99      0.91      2569
           5       0.00      0.00      0.00        65

    accuracy                           0.84      3057
   macro avg       0.66      0.41      0.40      3057
weighted avg       0.78      0.84      0.78      3057

CPU times: total: 5.22 s
Wall time: 6.42 s
```

Fig. 14. Classification report of Voting Classifier

Recall is a measure of the classifier's ability to identify all instances of a given class. It is the number of true positive predictions divided by the total number of instances of that class in the dataset.

The f1-score is the harmonic mean of precision and recall, and is a single metric that combines both precision and recall into a single score.

Support is the number of instances of each class in the dataset.

In this particular classification report, the classifier appears to perform well on class 4, with a high precision, recall, and f1-score. However, it performs poorly on classes 1, 3, and 5, with low precision, recall, and f1-scores. The classifier also appears to have only made one prediction for class 0, which was correct, resulting in a perfect precision, recall, and f1-score for that class.

7) Feature Importance: Feature importance indicates how much each feature contributes to the model's prediction. Basically, it determines the degree of usefulness of a specific variable for a current model and prediction.

Here, we have used the popularity column as the target feature that predicts the popularity of the merchandise. To predict popularity, two algorithms were used: logistic regression and random forest classifier. We got an accuracy of 84% in both models, but other comparisons show that the random forest has improved performance for classes 0 and 4.

The accuracy is unchanged, but the macro-average of precision and recall has improved to 0.42, and the Random Forest was chosen as the best method for classifying the popularity.

The store ratio is the most important data in the dataset. We used a random forest classifier for prediction.

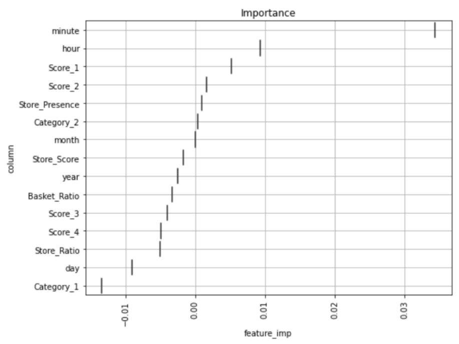


Fig. 15. Logistic Regression

As we can see in the random forest classifier graph, the store ratio has a higher value compared to logistic.

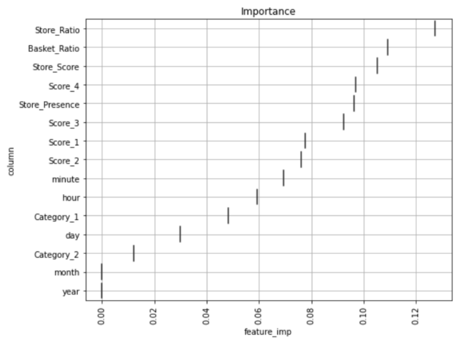


Fig. 16. Random Forest

IV. Comparison

Based on the classification reports it appears that the Random Forest Classifier has the best overall performance, followed by the Voting classifier, Logistic Regression, and KNN.

In terms of precision, the Random Forest classifier performs the best for all classes except for class 1, for which it has a precision of 1.00. The Voting classifier also has a high precision for all classes except for class 1, where it has a precision of 0.50. The Logistic Regression classifier has a precision of 0.00 for classes 0, 1, and 5, and a precision of 0.84 for class 4. The KNN classifier has a precision of 0.00 for classes 0, 1, and 5, and a precision of 0.84 for class 4, which is the same as the Logistic Regression classifier.

In terms of recall, the Random Forest classifier performs the best for all classes except for class 1, for which it has a recall of 0.02. The Voting classifier has a recall of 0.02 for class 1, a recall of 0.04 for class 3, and a recall of 0.99 for class 4. The Logistic Regression classifier has a recall of 0.00 for all classes, and the KNN classifier has a recall of 0.03 for class 3 and a recall of 0.98 for class 4.

In terms of f1-score, the Random Forest classifier has the highest f1-score for all classes except for class 1, where it has a score of 0.04. The Voting classifier has a f1-score of 0.04 for class 1, a score of 0.07 for class 3, and a score of 0.91 for class 4. The Logistic Regression classifier has a f1-score of 0.00 for all classes, and the KNN classifier has a f1-score of 0.06 for class 3 and a score of 0.91 for class 4.

Overall, the Random Forest classifier appears to be the best performing model , as it has the highest precision, recall, and f1-score for most classes. However, it is important to note that the performance of a classifier can depend on a variety of factors such as the quality of the data, the appropriate choice of model and hyperparameters, and the specific application. It is always a good idea to consider the specific requirements and goals of the task at hand when choosing a classifier.

V. Future Directions

We have interpreted that we should expand our dataset to contain more features that can best describe the target variable, i.e. Store ratio for predicting the popularity. This is what we would want to implement in the future. In order to grasp the many methods that others have employed to solve the problem, we will research past solutions that have been applied.

VI. Conclusion

We have used a variety of EDA techniques to deeply understand the data. According to the classification reports, the Random Forest classifier, followed by the Voting classifier, Logistic Regression, and KNN, appears to have the best overall performance. Achieved an accuracy of 84%. In the case of popularity predictions, it predicted that the most important factor in determining the popularity of merchandise is the "store ratio" attribute. The Random Forest Classifier provided us with the highest test result among the other algorithms we used.

References

- Osiris " Machine Hack ML Merchandise Popularity Prediction" kaggle competition.
- Moutinho, Luiz, and Graeme D. Hutcheson. "Store patronage: the utility of a multi-method, multi-nomial logistic regression model for predicting store choice."

Portuguese Journal of Management Studies 11.1 (2006): 5-26.

- Speiser, Jaime Lynn, et al. ”A comparison of random forest variable selection methods for classification prediction modeling.” Expert systems with applications 134 (2019): 93-101.