Machine Learning Project Report

Predict Mobile App Success

Milestone 1: Regression

Mobile applications are important in the present and future and used in daily basis. Evaluating mobile app success help developers to decide which variables to take into consideration. In this report we explain how our "Predict Mobile App Success" project works.

Preprocessing Techniques

To train a model on data, it must be clean and ready-to-use. So, various pre-processing techniques are applied to "AppleStore_training" dataset before passing it to the model.

1. One-Hot-Encoding
   One of the most famous methods in categorial encoding. It split the column containing N discrete values into N binary columns consisting of 0's and 1's. We used this technique to encode "prime genre" column.
   It is implemented by sklearn preprocessing functions: ColumnTransformer, OneHotEncoder and fit_transform.

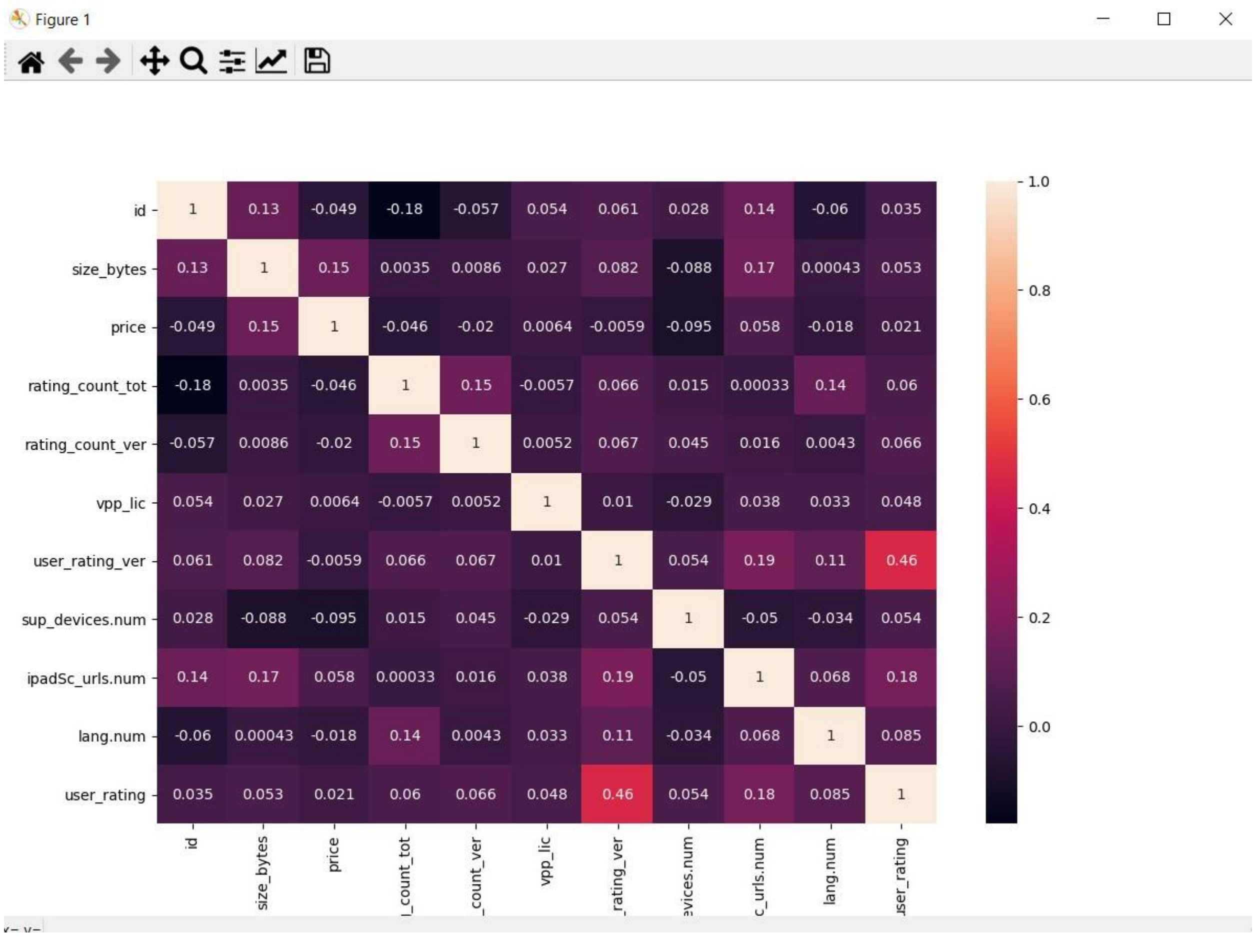
2. Remove Columns
   Since not all columns are needed in the training process, we discarded columns that are unnecessary like "currency" and "ver".
   This is performed by "del" function.

3. Remove Rows
   Datasets often contain empty values that can cause problems for machine learning models. In order to deal with this issue, rows that have empty entries are dropped.
   This is performed by" dropna" function.

4. Dealing with signs and numbers
   Column like “content rating” contains numbers and signs values like “4+, 12+, and so on”. Humans understands the meanings behind these values as the appropriate age, but the machine interprets them as strings. To make the machine understand them as numbers we removed the plus sign from the values.
   This is done by looping on column values and deleting the plus sign.

Dataset Analysis

Data correlation is a way to understand the relationship between multiple variables and attributes in dataset.



	id	size_bytes	price	_count_tot	_count_ver	vpp_lic	_rating_ver	evices.num	c_urls.num	lang.num	ser_rating
id	1	0.13	-0.049	-0.18	-0.057	0.054	0.061	0.028	0.14	-0.06	0.035
size_bytes	0.13	1	0.15	0.0035	0.0086	0.027	0.082	-0.088	0.17	0.00043	0.053
price	-0.049	0.15	1	-0.046	-0.02	0.0064	-0.0059	-0.095	0.058	-0.018	0.021
rating_count_tot	-0.18	0.0035	-0.046	1	0.15	-0.0057	0.066	0.015	0.00033	0.14	0.06
rating_count_ver	-0.057	0.0086	-0.02	0.15	1	0.0052	0.067	0.045	0.016	0.0043	0.066
vpp_lic	0.054	0.027	0.0064	-0.0057	0.0052	1	0.01	-0.029	0.038	0.033	0.048
user_rating_ver	0.061	0.082	-0.0059	0.066	0.067	0.01	1	0.054	0.19	0.11	0.46
sup_devices.num	0.028	-0.088	-0.095	0.015	0.045	-0.029	0.054	1	-0.05	-0.034	0.054
ipadSc_urls.num	0.14	0.17	0.058	0.00033	0.016	0.038	0.19	-0.05	1	0.068	0.18
lang.num	-0.06	0.00043	-0.018	0.14	0.0043	0.033	0.11	-0.034	0.068	1	0.085
user_rating	0.035	0.053	0.021	0.06	0.066	0.048	0.46	0.054	0.18	0.085	1

Figure 1. Data correlation between features.

The correlation plot displays the relationship between “AppleStore_training” dataset features as shown in Figure 1.

Regression Techniques

Three regression models are used in this project.

1. Multiple linear regression.
2. Polynomial regression. (with polynomial degree = 4)
3. Lasso regression. (with alpha = 0.1)

Models Comparison

Model	Multiple	Polynomial	Lasso
Intercept	2.7184589189137265	4.071641734662759	3.409244496367744
Mean Square Error	0.4474345135491682	0.5983901067569111	0.471573889254434
Example: True value	4.0	4.0	4.0
Example: Predicted value	4.17802830103025	4.071643962043115	4.115013478932136
Training Time	0.05186 second	38.5931994 second	0.038896 second

Figure 2. Comparison table between the 3 models.

The table above shows the interception, mean square error, user rating value from the dataset, the predicted user rating value and finally the training time for each model.

Features Used/Discarded

The dataset contains multiple features, but not all of them are used in our models, here is the used and discarded features list.

Used features:

- size_bytes
- price
- rating_count_tot
- rating_count_ver
- vpp_lic
- user_rating_ver
- ver
- cont_rating
- prime_genre (used in one-hot-encoding then discarded)
- sup_devices.num
- ipadSc_urls.num
- lang.num

Discarded features:

- id
- track_name
- currency
- ver

Training/Testing Sets Size

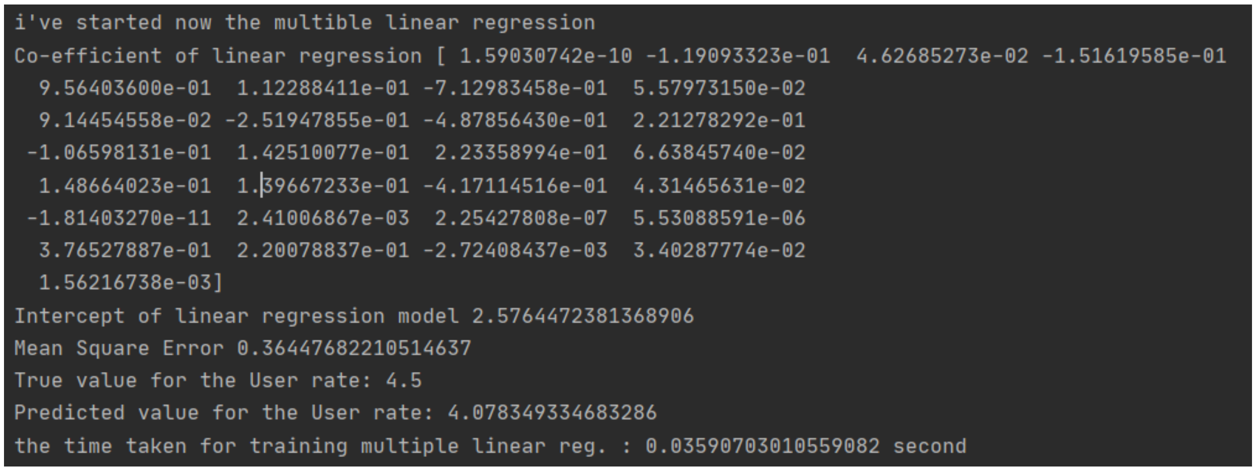
Initially all dataset rows are loaded, then the rows containing empty values are discarded, then the data is split into 70% training set and 30% testing set.

We shuffled the data, so the results (MSE) are slightly different each run.

Further Techniques to Improve Results

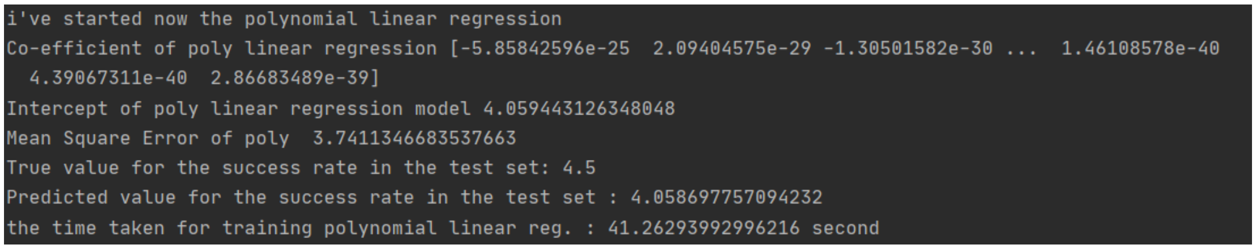
Lasso regression.

Screenshots



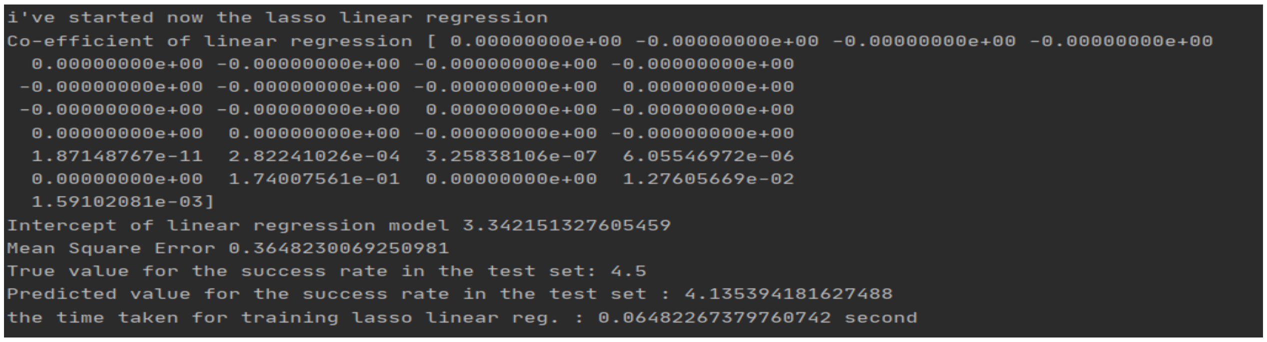
```
i've started now the multible linear regression
Co-efficient of linear regression [ 1.59030742e-10 -1.19093323e-01  4.62685273e-02 -1.51619585e-01
  9.56403600e-01  1.12288411e-01 -7.12983458e-01  5.57973150e-02
  9.14454558e-02 -2.51947855e-01 -4.87856430e-01  2.21278292e-01
 -1.06598131e-01  1.42510077e-01  2.23358994e-01  6.63845740e-02
  1.48664023e-01  1.39667233e-01 -4.17114516e-01  4.31465631e-02
 -1.81403270e-11  2.41006867e-03  2.25427808e-07  5.53088591e-06
  3.76527887e-01  2.20078837e-01 -2.72408437e-03  3.40287774e-02
  1.56216738e-03]
Intercept of linear regression model 2.5764472381368906
Mean Square Error 0.36447682210514637
True value for the User rate: 4.5
Predicted value for the User rate: 4.078349334683286
the time taken for training multiple linear reg. : 0.03590703010559082 second
```

Figure 3. Multiple linear regression results.



```
i've started now the polynomial linear regression
Co-efficient of poly linear regression [-5.85842596e-25  2.09404575e-29 -1.30501582e-30 ...  1.46108578e-40
  4.39067311e-40  2.86683489e-39]
Intercept of poly linear regression model 4.059443126348048
Mean Square Error of poly  3.7411346683537663
True value for the success rate in the test set: 4.5
Predicted value for the success rate in the test set : 4.058697757094232
the time taken for training polynomial linear reg. : 41.26293992996216 second
```

Figure 4. Polynomial linear regression results.



```
i've started now the lasso linear regression
Co-efficient of linear regression [ 0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  0.00000000e+00 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00
 -0.00000000e+00 -0.00000000e+00 -0.00000000e+00  0.00000000e+00
 -0.00000000e+00 -0.00000000e+00  0.00000000e+00 -0.00000000e+00
  0.00000000e+00  0.00000000e+00 -0.00000000e+00 -0.00000000e+00
  1.87148767e-11  2.82241026e-04  3.25838106e-07  6.05546972e-06
  0.00000000e+00  1.74007561e-01  0.00000000e+00  1.27605669e-02
  1.59102081e-03]
Intercept of linear regression model 3.342151327605459
Mean Square Error 0.3648230069250981
True value for the success rate in the test set: 4.5
Predicted value for the success rate in the test set : 4.135394181627488
the time taken for training lasso linear reg. : 0.06482267379760742 second
```

Figure 5. Lasso linear regression results.

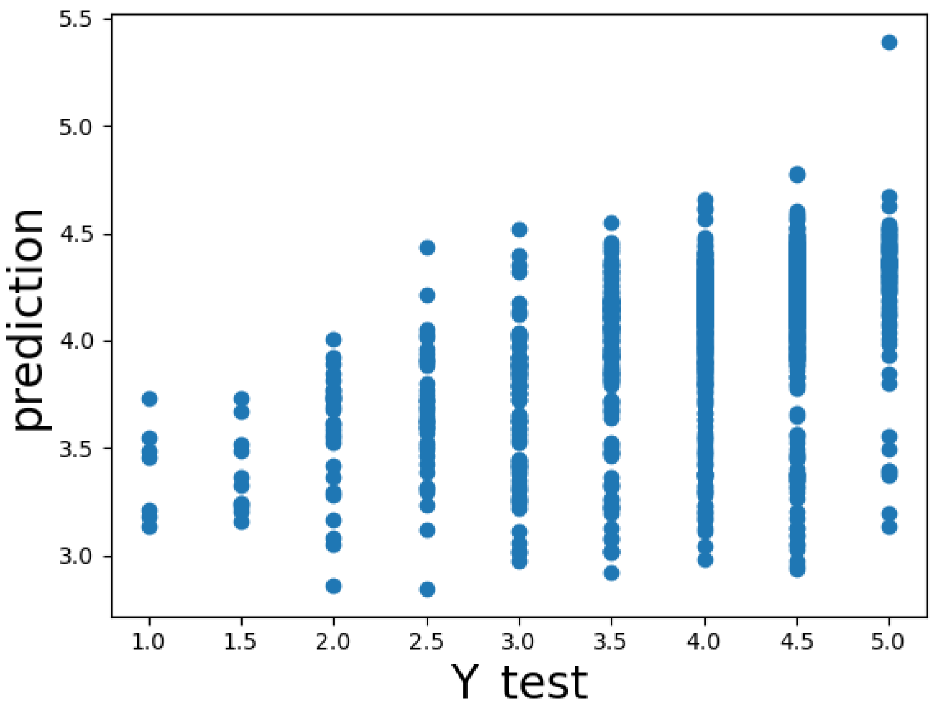


Figure 6. Multiple linear regression plot.

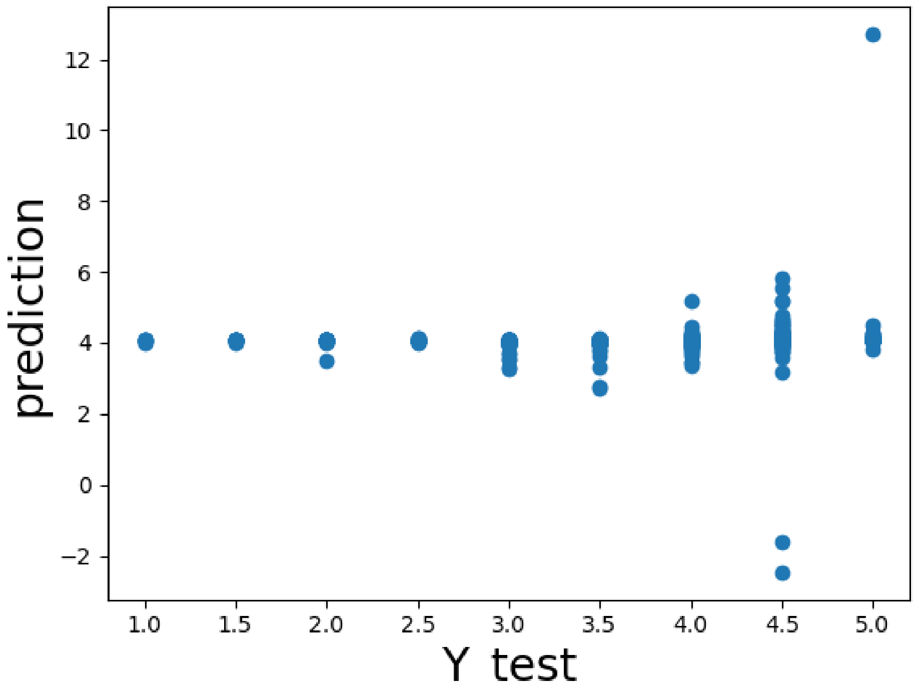


Figure 7. Polynomial regression plot.

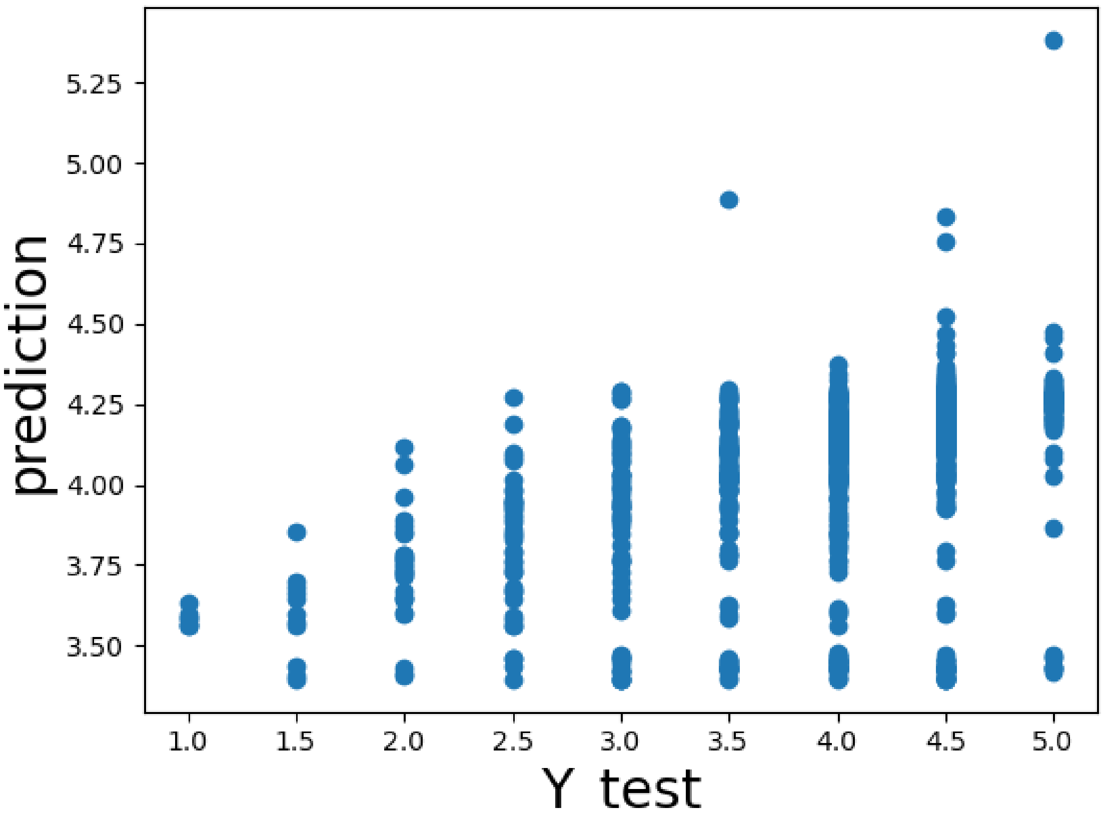


Figure 8. Lasso linear regression plot.

Note:

Since the data has so many features, it is impossible to represent in 2-D or 3-D plots, so we plotted the y test values against our predicted values to display the results of our models instead.

Conclusion

Regression techniques sets up an equation to explain the significant relationship between one or more features and response variables and also to estimate current observations.

Multiple linear regression looks at the relationships within a bunch of information. Instead of just looking at how one thing relates to another thing (simple linear regression).

Polynomial regression technique is used to execute a model that is fit to manage non-linearly separated data. The best-fitted line is not a straight line, instead, a curve that best-fitted to data points.

Regularization refers to a broad range of techniques for artificially forcing your model to be simpler and prevent overfitting. Lasso Regression adds “absolute value of magnitude” of coefficient as penalty term to the loss function.

Milestone 2: Classification

Preprocessing and feature selection

The same preprocessing techniques in the regression phase are applied in the classification phase, which include: 1- One hot encoding for the prime_genre column. 2- Removing id, track_name, currency and ver columns. 3- Removing rows with empty fields. 4- Deleting the plus sign in cont_rating column.

In addition to the previous 4 techniques, 2 more are added for classification.

1. Feature scaling
   In order to improve the classification accuracy, feature scaling is applied to the features. This is done by Standard Scaler from sklearn.preprocessing library.
2. Transforming target labels to numbers
   Since the data values cannot be strings, the target column "rate" labels are transformed to numbers to make the machine learning models understand them better. The labels are translated as follows: 1 for "High" class, 0 for "Intermediate" class and -1 for "Low" class.

Classification Models Summary

To visualize and understand the difference between our classification models better, the following graphs illustrate the classification accuracy, training time and testing time for each classification model respectfully.

Note:
Since the shuffle parameter in the train test split function is equal to True, the data points used in training and testing differ each run, which results in slight change in the results. So, the following graphs are not constants.

1. Classification accuracy

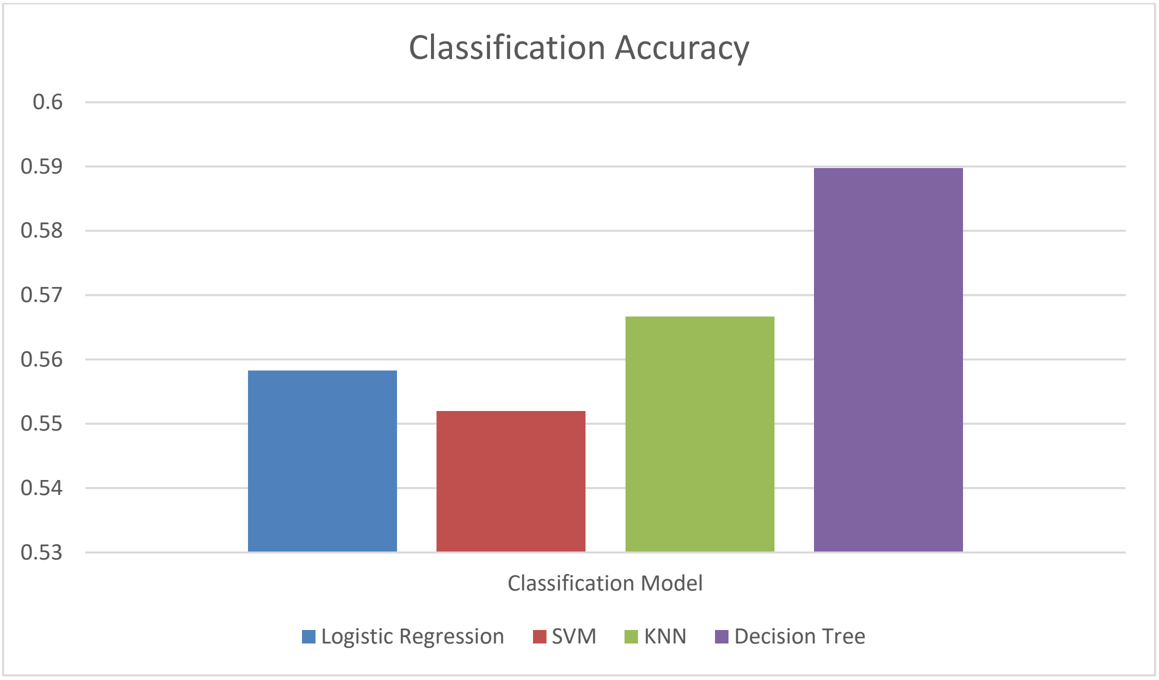


Figure 9. Classification accuracy bar graph.

2. Total training time

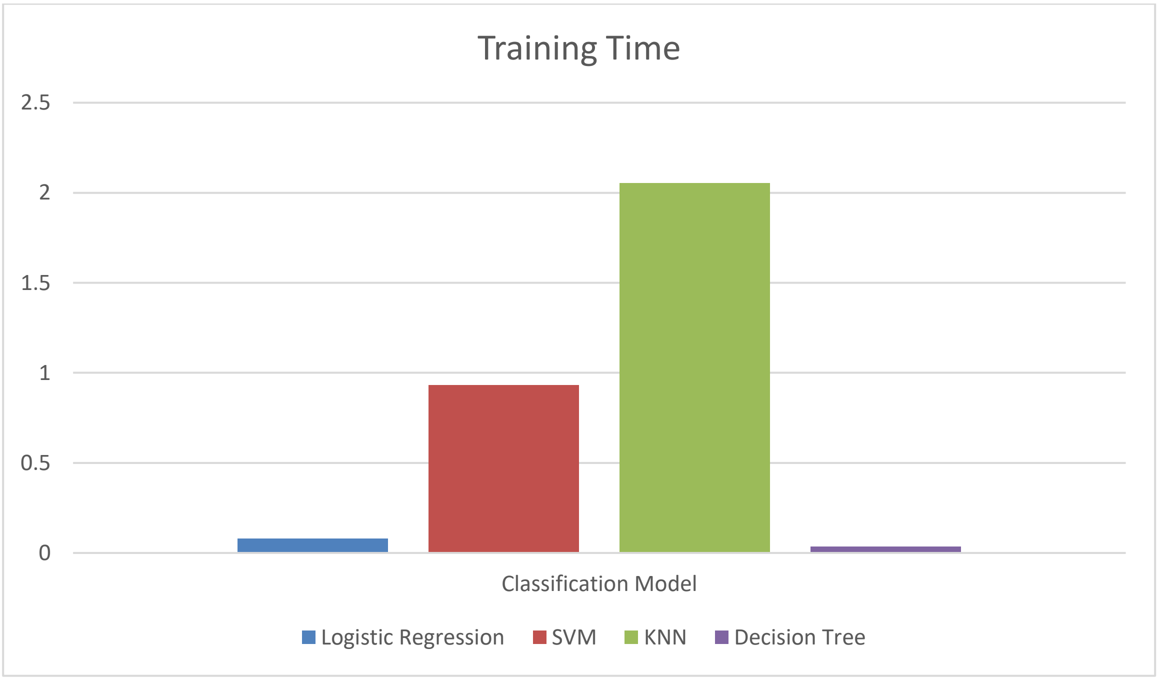


Figure 10. Classification models training time bar graph.

3. Total test time

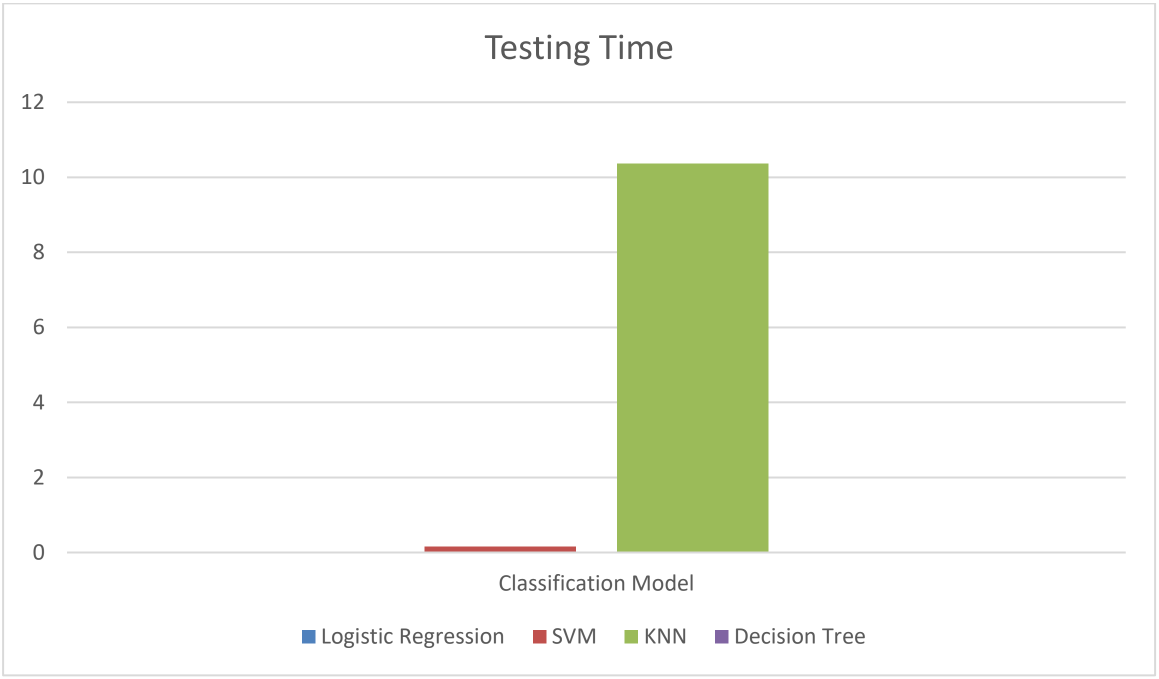


Figure 11. Classification models testing time bar graph.

Hyperparameter Tuning

Changing the hyperparameters values, change the performance of the classification model. Here are the details of the 4 models we used and their results with different hyperparameter values.

1. Logistic Regression model

   The following is a simple run showing the effect of changing the regularization parameter C:

```
#
Logistic reg (C=0.000001) training took 0.051023 seconds
Logistic reg (C=0.000001) testing took 0.000000 seconds
logistic reg (C=0.000001) accuracy: 0.49737670514165794
#
Logistic reg (C=0.1) training took 0.081383 seconds
Logistic reg (C=0.1) testing took 0.000998 seconds
logistic reg (C=0.1) accuracy: 0.5508919202518363
```

```
#
Logistic reg (C=10) training took 0.171032 seconds
Logistic reg (C=10) testing took 0.000000 seconds
logistic reg (C=10) accuracy: 0.5508919202518363
#
```

By observing the results, it is clear that the very small value (C=0.000001) took the minimum time in training but resulted in the lowest accuracy, and the large value (C=10) took the maximum training time, but the accuracy is nearly the same as (C=0.1). In addition to that, (C=10) gave the warning that the algorithm does not converge, which means that the algorithm likely reached the solution.

Note: the other hyperparameter (Penalty= 'l2') cannot be changed since the logistic regression is implemented by ‘lbfgs’ solver which supports only l2 penalty.

2. SVM model

   The following is a simple run showing the effect of changing the regularization parameter C:

```
#
SVM (C=0.000001) training took 0.940482 second
SVM (C=0.000001) testing took 0.176559 second
SVM (C=0.000001) test accuracy: 0.5120671563483735
SVM (C=0.000001) train accuracy: 0.5184990816058778
#
SVM (C=0.1) training took 0.999298 second
SVM (C=0.1) testing took 0.166588 second
SVM (C=0.1) test accuracy: 0.565582371458552
SVM (C=0.1) train accuracy: 0.5791130936762005
#
SVM (C=1) training took 1.103022 second
SVM (C=1) testing took 0.157577 second
SVM (C=1) test accuracy: 0.5645330535152151
SVM (C=1) train accuracy: 0.5859354500131199
```

```
#
SVM (C=100) training took 1.539880 second
SVM (C=100) testing took 0.149596 second
SVM (C=100) test accuracy: 0.5676810073452256
SVM (C=100) train accuracy: 0.6720020991865652
#
```

It is known that large C leads to overfitting, and small C leads to underfitting.
The results above show that C with very small value gives the lowest accuracy in both training and testing, and C with very large value gives the highest training accuracy but the testing accuracy is close to its value when C=0.01 and C=1 which means it did not change very much since then which is a symptom for overfitting.
Regarding the training time, it increases as C values increase, however, the testing time decreases as C values increase.

3. KNN model
   The KNN model does not have hyperparameters to tune, so we tried different values (from 1 to 50) for K instead and observed the error for each KNN model.

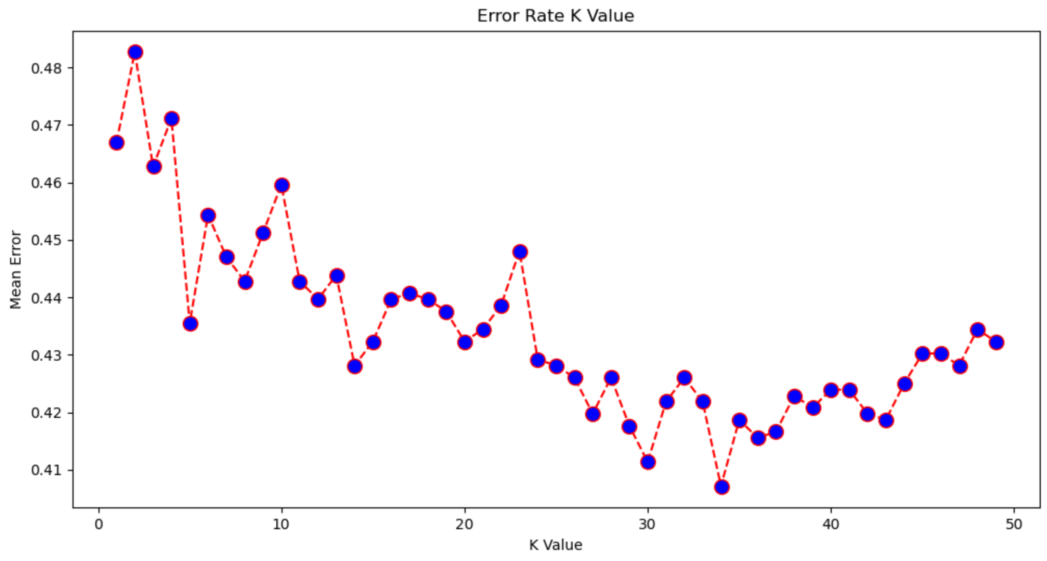


Figure 12. K values in range [1-50] against error plot.

Note: since the data is shuffled, the K-error plot changes every run.

Here is a simple run displaying the KNN results:

```
#
KNN training took 1.960643 second
KNN testing took 9.943888 second
KNN accuracy: 0.5928646379853095
#
```

4. Decision Tree Classifier model

   The following is a simple run showing the effect of changing the maximum depth value:

```
#
Decision Tree (max_depth=1) training took 0.080785 second
Decision Tree (max_depth=1) testing took 0.002986 second
Decision Tree (max_depth=1) accuracy: 0.5718782791185729
#
Decision Tree (max_depth=3) training took 0.011968 second
Decision Tree (max_depth=3) testing took 0.001084 second
Decision Tree (max_depth=3) accuracy: 0.5949632738719832
#
Decision Tree (max_depth=100) training took 0.040802 second
Decision Tree (max_depth=100) testing took 0.000069 second
Decision Tree (max_depth=100) accuracy: 0.5267576075550892
#
```

   When a large value is assigned to the maximum depth, this leads the model to overfit. The accuracy has improved when max_depth=3, then it dropped when max_depth=100, which means that overfitting happened.
   The training and testing time both have small values. When the maximum depth increases, the training time also increases.

Conclusion

Classification is used when the target classes are categorical. To improve the performance of the classification models, feature scaling can be applied.

Hyperparameter tuning is important for the classification accuracy and preventing overfitting and underfitting.

By observing our classification models, the decision tree model gave the best results in the accuracy, training time and testing time.

Followed by it, the KNN gave the second-best accuracy, however, it has the worst training and testing time.

Finally, came the logistic regression then the SVM.