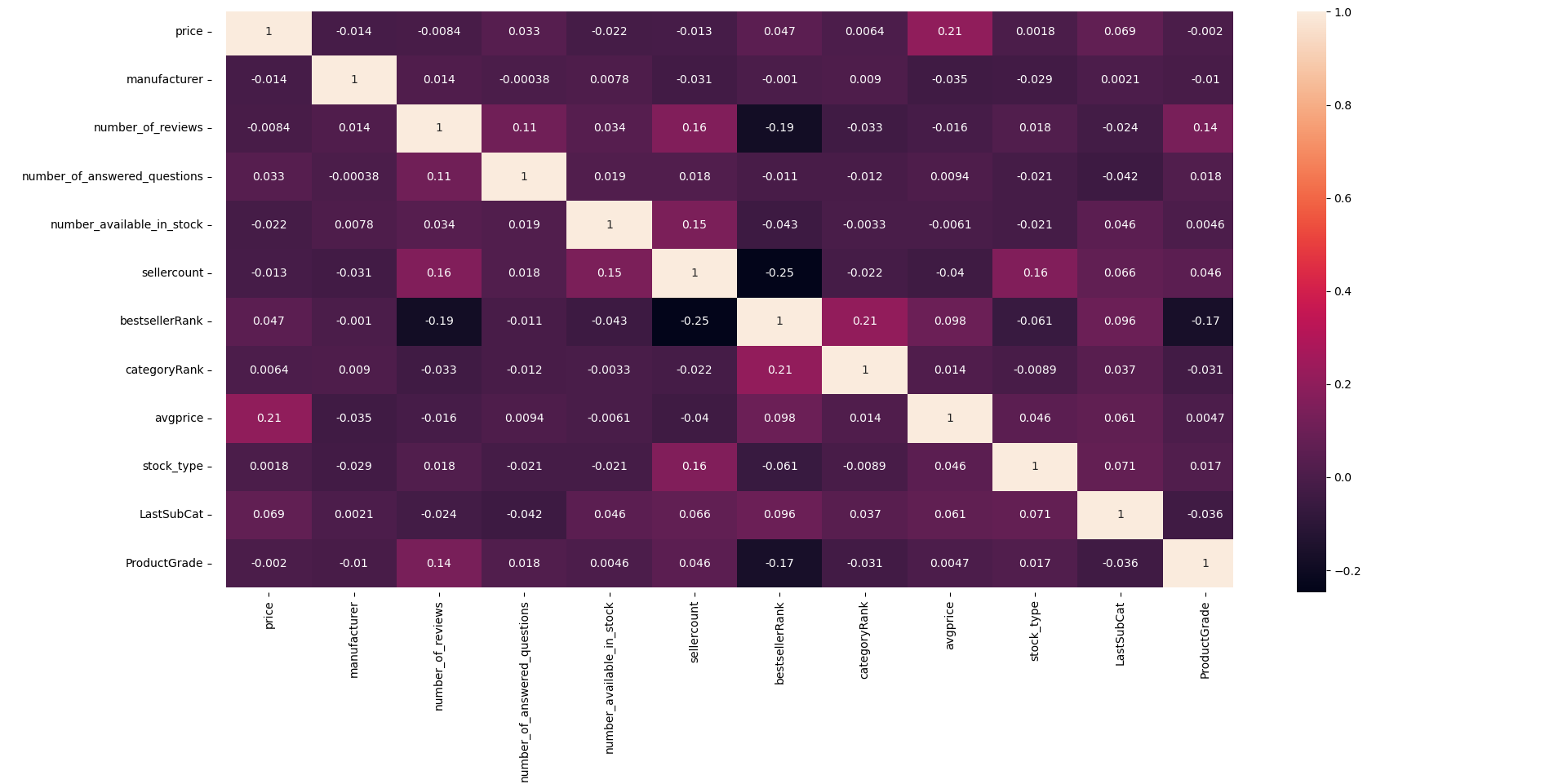
|  |  |
| --- | --- |
|  |  |

Page 1

**Same as MileStone1**

Page 4

* After calculating correlation and plotting the heatmap to see the most correlated features, we decided to take the Top 4% Correlation training features with the Product Grade column as the highest correlation absolute value with the Product Grade column is 0.17 (Best seller rank column) as it is shown in the screenshot below:
* According to the heatmap and the threshold value (0.04) for selecting variables we can see that the variables which have higher correlation coefficient values with the Product Grade are best seller rank, number of reviews and seller count. So, we are going to use these 3 variables for our prediction model.

* Logistic Regression:

Hyperparameters Tuning in Model 1:

We have chosen 4 hyperparameters to vary:

* C: Inverse of regularization strength, smaller values specify stronger regularization.

Values we used [100, 1.0, 0.1, 0.01]

We have noticed that when C has bigger value the accuracy will increase while

All the other hyperparameters are fixed. (When C is high it will classify all the data points correctly, also there is a chance to overfit).

* Solver: Algorithm to use in the optimization problem.

For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss.

‘liblinear’ is limited to one-versus-rest schemes.

Values we used ['newton-cg', 'lbfgs','sag']

We have noticed that changing the solver value doesn’t affect the accuracy value.

* Multiclass: Values we used [‘multinomial’,’ovr’]

‘multinomial’ is unavailable when solver=’liblinear’.

Multinomial option gives higher accuracy values than ovr.

* Penalty: Values we used [ L1, L2]

Some penalties may not work with some solvers. As L1 only works with liblinear solver

So, we have studied the effect of the penalty hyperparameter only when the solver is liblinear and the multiclass is ovr (As ‘liblinear’ is limited to one-versus-rest schemes).

L1 gives higher accuracy values than L2.

Best Accuracy is: 65.760452

Achieved using {'C': 100, 'multi\_class': 'multinomial', 'solver': 'newton-cg'}.

* Training Accuracy: 0.657554379776602
* Training Time in seconds: 0.24170732498168945
* Testing Accuracy: 0.6407995296884186
* Testing Time in seconds: 0.008045196533203125
* cross validation score with k=10 (10-fold cross validation): 0.6576045175779563
* Support Vector Machine:

We have trained SVM on the training dataset with all the kernel functions and we got these results C=100:

* Testing Accuracy using linear kernel (one vs rest): 0.631981187536743
* Testing Accuracy using linear kernel (one vs one): 0.6390358612580834
* Testing Accuracy using RBF kernel (one vs rest): 0.6378600823045267
* Testing Accuracy using RBF kernel (one vs one): 0.6407995296884186
* Testing Accuracy for polynomial degree 3 (one vs one):

So, the RBF kernel (one vs one) is the most appropriate kernel function as it has the highest accuracy.

Hyperparameters Tuning in Model 2:

We have chosen 2 hyperparameters to vary in RBF Kernel (one vs one):

* C: C is the penalty parameter, which represents misclassification or error term. This is how you can control the trade-off between decision boundary and misclassification term. Values we used [100, 1.0, 0.1]

We have noticed that when C has bigger value the accuracy will increase while

All the other hyperparameters are fixed (when C is high it will classify all the data points correctly, also there is a chance to overfit).

* Gamma**:** It defines how far influences the calculation of plausible line of separation. Values we used [1, 0.1, 0.01, 0.001, 0.0001].

when gamma is higher, nearby points will have high influence thus higher accuracy; low gamma means far away points also be considered to get the decision boundary.

Best Accuracy is: 0.64079953

Achieved using {'C': 100, gamma:0.1, kernel: RBF}.

* Testing Accuracy for RBF (one vs one): 0.64079953
* Training Time in seconds RBF (one vs one): 7.615013122558594
* Testing Time in seconds RBF (one vs one): 3.0604875087738037
* cross validation score with k=10 (10-fold cross validation): 0.6592721200080621
* Decision tree classifier:

We have trained Decision tree classifier on the training dataset once using Adaboost and another without and we got these results:

* Testing Accuracy using Decision tree classifier: 64.55026455026454
* Testing Accuracy using Adaboost: 60.258671369782476

So, the Decision tree classifier has the highest accuracy.

Hyperparameters Tuning in Model 3:

We have chosen 3 hyperparameters to vary in Decision Tree Classifier:

* C: Inverse of regularization strength, smaller values specify stronger regularization.

Values we used [100, 1.0, 0.1, 0.01]

We have noticed that when C has bigger value the accuracy will increase while

All the other hyperparameters are fixed. (When C is high it will classify all the data points correctly, also there is a chance to overfit).

* Solver: Algorithm to use in the optimization problem.

For multiclass problems, only ‘newton-cg’, ‘sag’, ‘saga’ and ‘lbfgs’ handle multinomial loss.

‘liblinear’ is limited to one-versus-rest schemes.

Values we used ['newton-cg', 'lbfgs','sag']

We have noticed that changing the solver value doesn’t affect the accuracy value.

* Multiclass: Values we used [‘multinomial’,’ovr’]

‘multinomial’ is unavailable when solver=’liblinear’.

Multinomial option gives higher accuracy values than ovr.

* Penalty: Values we used [ L1, L2]

Some penalties may not work with some solvers. As L1 only works with liblinear solver

So, we have studied the effect of the penalty hyperparameter only when the solver is liblinear and the multiclass is ovr (As ‘liblinear’ is limited to one-versus-rest schemes).

L1 gives higher accuracy values than L2.

Best Accuracy is: 65.760452

Achieved using {'C': 100, 'multi\_class': 'multinomial', 'solver': 'newton-cg'}.