Shahid Beheshti University

Faculty of Mathematical Science

Department of Computer Science

Fundamentals of Machine Learning– Spring 2022 – Assignment 1 – Part 1

Mobile Price Classification

By:

Arman Davoodi

Abstract

This report gives an overview of different statistical analyses made on a dataset of mobile data consisting of 20 features. Moreover, this report describes how a Logistic Regression classifier was used to predict the price range of any given mobile device and the data preprocessing steps done, to prepare the data for training the model.

Dataset

The dataset referred to in this report is obtained from the Kaggle. This data set has 6 Boolean, 1 categorical, 11 integer and 2 floating-point features. For each mobile device, there is also a price range column which is categorical data consisting of 4 values. This column is also the target of our classification problem. Also, the dataset has 2000 records. Additional information about each feature can be found on the source. Numerical features distributions are shown in *figure 1* and histograms of categorical and binary data types are illustrated in *figure 2*. In addition, some data aggregators for each feature are presented in *table 1*.

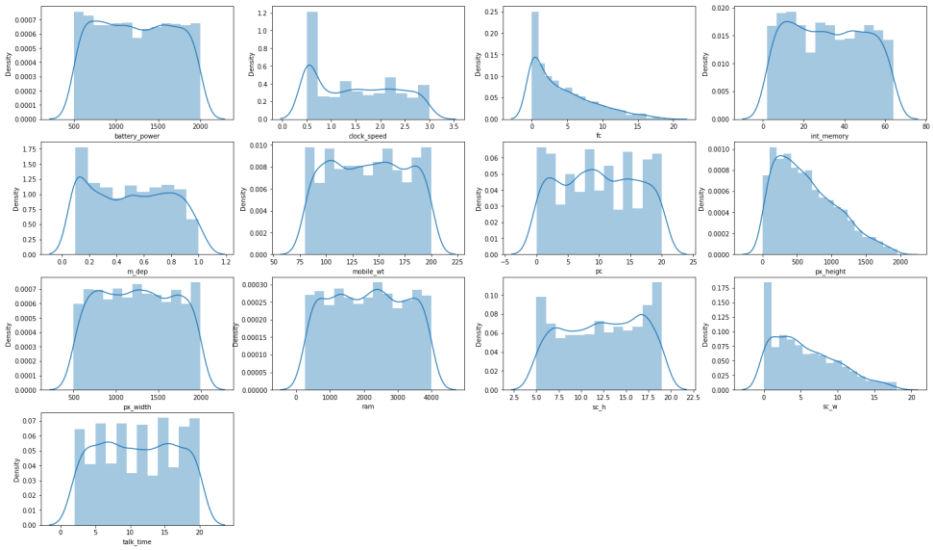

Figure 1

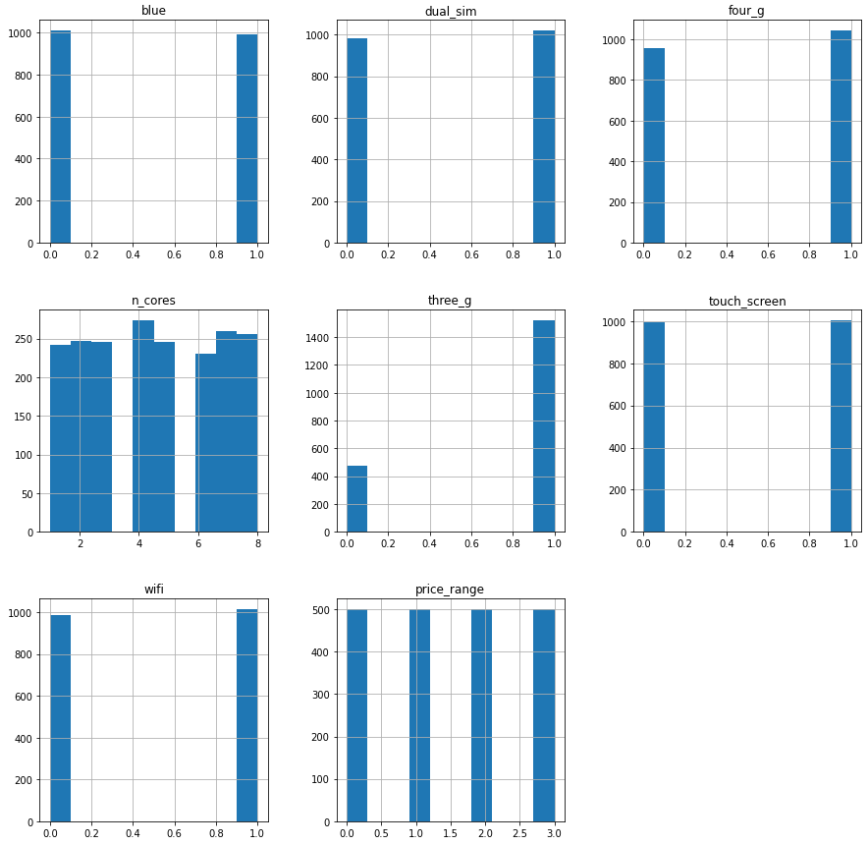

Figure 2

Table 1

	battery power	blue	clock speed	dual sim	fc	4G	int memory	m dep	mobile wt	n cores	pc	px height	px width	ram	sc h	sc w	talk time	3G	touch screen	wifi	Price range
count	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000	2000
mean	1238.5	0.495	1.522	0.509	4.309	0.521	32.046	0.502	140.25	4.52	9.916	645.11	1251.5	2124.2	12.31	5.767	11.011	0.761	0.503	0.507	1.5
std	439.4	0.500	0.816	0.500	4.341	0.500	18.146	0.288	35.400	2.288	6.064	443.78	432.20	1087.7	4.21	4.356	5.464	0.426	0.500	0.500	1.118
min	501	0	0.5	0	0	0	2	0.1	80	1	0	0	500	256	5	0	2	0	0	0	0
25%	851.75	0	0.7	0	1	0	16	0.2	109	3	5	282.75	874.75	1207.5	9	2	6	1	0	0	0.75
50%	1226	0	1.5	1	3	1	32	0.5	141	4	10	564	1247	2146.5	12	5	11	1	1	1	1.5
75%	1615.25	1	2.2	1	7	1	48	0.8	170	7	15	947.25	1633	3064.5	16	9	16	1	1	1	2.25
max	1998	1	3	1	19	1	64	1.0	200	8	20	1960	1998	3998	19	18	20	1	1	1	3

Outliers Detection and Removal

Two methods for outlier removal were considered in this task. The first method was Inter-Quartile Range (IQR) proximity rule which was used for features with the skewed distribution shown in *figure 1*. In this algorithm, the records which fall below $Q_1 – 1.5$ IQR or above $Q_3 + 1.5$ IQR are considered to be outliers where Q_1 and Q_3 are respectively the 25^{th} and 75^{th} percentile of the dataset, and IQR represents the inter-quartile range and is given by $Q_3 – Q_1$.

The other outlier removal method was the percentile-based approach. This technique works by setting a particular threshold value like α, which is decided based on the problem statement. Then the data points higher than $(100-\alpha)^{th}$ and lower than α^{th} percentiles are removed from our dataset.

However, for detecting the outliers the boxplot of each feature was plotted. The boxplot of skewed and non-skewed distributed features are demonstrated in *figure 3* and *figure 4* respectively.

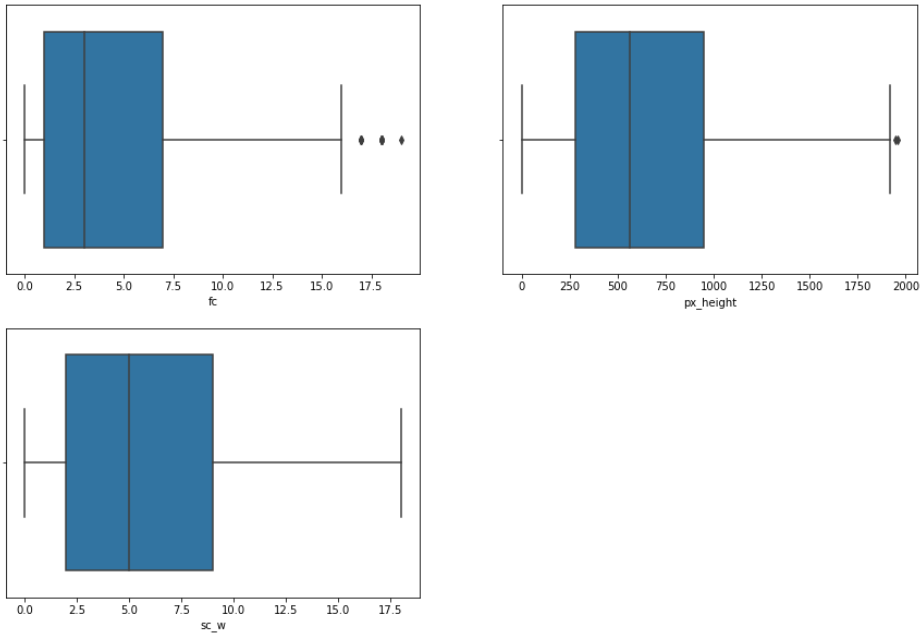

Figure 3

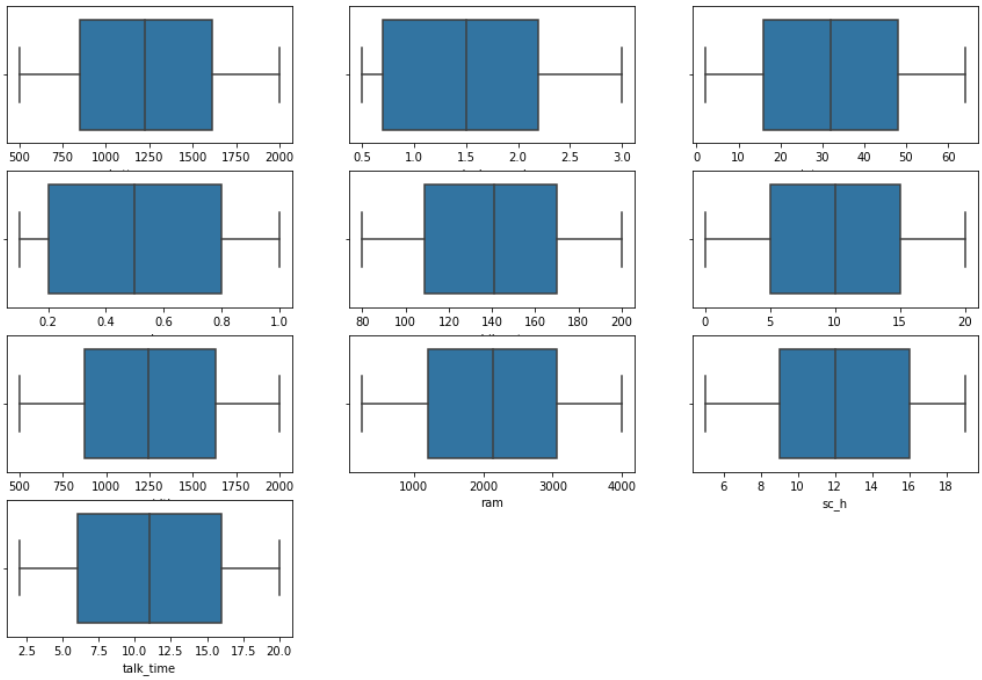

Figure 4

As it can be seen from *figure 3* and *figure 4* only fc and px_height have outlier data and since both of these features have skewed distribution, the IQR method is used to remove their outliers. The boxplot of these two features after applying the IQR outlier removal method on them is illustrated in *figure 5*.

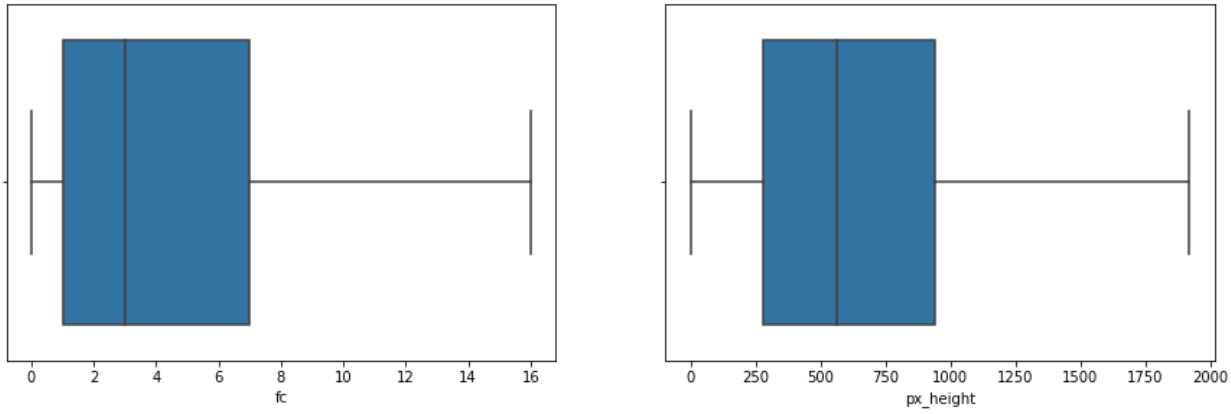

Figure 5

In the end, after applying outlier removal to the dataset, 20 records were considered to be outliers and were removed from the dataset.

Data Analysis

To get a general idea of the relationship between every two features and the Price Range, a scatter plot for each pair of features is plotted where each point represents a record of our dataset and the colours of each point represents the price range of that record with the darkest and palest colors representing the highest (3) and the lowest (0) price ranges respectively. These plots are shown in *figure 6* (this figure is also saved as MPC_pair_plot.png in the respective git repository).

Figure 6

As it can be seen in *figure 6*, the ram feature can distinguish different price range classes in a linear manner. This can also be seen by computing the correlation between ram and price range which is approximately 0.917.

Also the feature pairs pc and fc, four_g and three_g, px_height and px_width, and sc_h and sc_w have the highest correlations after ram-price_range pair which are 0.636, 0.585, 0.506 and 0.504 respectively.

Hypothesis Testing

In this analysis, five hypothesis tests were performed.

Test1: Mobiles with price range 3 have more than 2500MB ram

By looking at the scatter plot of ram-price_range and the ram distribution of records with the highest price range illustrated in *figure 7* and *figure 8* respectively, we can see that this hypothesis may be true.

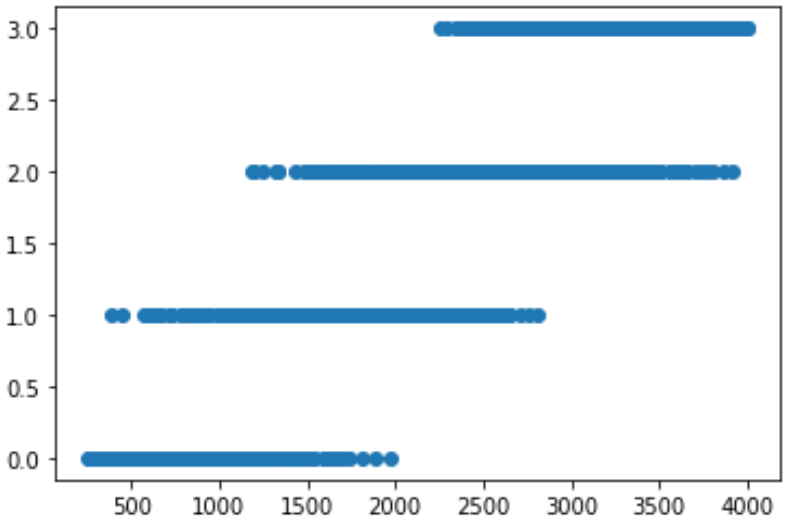

Figure 7: the x axis represents ram and the y axis represents price range

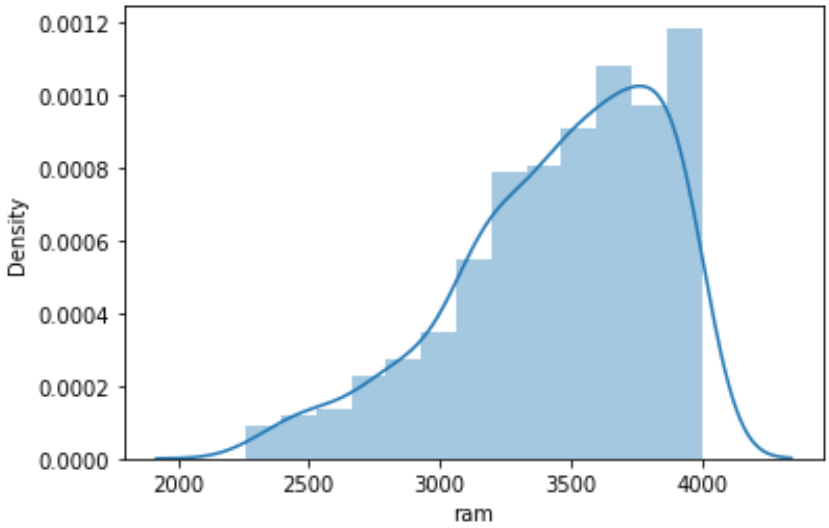

Figure 8

For testing this hypothesis we consider our null hypothesis or H_0 to be $\mu = 2500$ and our H_1 to be $\mu > 2500$ and we use the one-sample t-test to evaluate whether the hypothesis is true or not.

By doing this we get a t-stat equal to 53.76 and a p-value small enough to be considered 0 therefore our null hypothesis is rejected.

Test2: Clock speed is related to the number of cores

In this example, we consider our H_0 to be that the mean clock speed for any number of cores is the same. By using the one-way ANOVA test we get the f stat equal to 0.999 and a p-value equal to 0.429 which is higher than 0.05. Therefore, our null hypothesis is accepted.

Test3: There is a relationship between having 3G and 4G

By looking at the contingency table for these two features represented in *table 2* we can see that there is no mobile that has 3G but does not have 4G and most of the phones have both 3G and 4G.

Table 2

3G \ 4G	Doesn't have	Have
Doesn't have	474	0
Have	475	1031

By applying the Chi2 test we get a Chi2 stat of 674.29 and a p-value small enough to be considered 0 and therefore we can reject the null hypothesis and say that there is a relationship between having 3G and 4G.

Test4: pc and fc have monotonic relationship

By using spearman correlation we get a stat of 0.651 and a p-value of almost 0 therefore we can conclude that these two features are dependent.

Test5: px_height and px_width have monotonic relationship

By using spearman correlation we get a stat of 0.464 and a p-value of almost 0 therefore we can conclude that these two features are dependent.

Model

Considering the linear relationship between ram and our target combined with the fact that most of our features are numerical, Multinomial Logistic Regression Classifier can be a good model for our problem.

However, to get the best results for this model, various scaling methods on our dataset combined with different solvers and penalty norms for the model were used.

The dataset was used in its non-scaled, min-max scaled, standardized and robust scaled forms.

Also, newton-cg, lbfgs, sag and saga were each used as solvers. Moreover, for all solvers, it was checked whether is it better to not use a penalty norm at all or to use l2 as a penalty norm when the classifier is at its learning steps. For saga solver, l1 and elasticent penalty with l1-ratio of 0.5 were also checked.

For evaluating the performance of the model 20 percent of the data were used as tests and the rest were used to train the model.

Ultimately standardized data combined with saga solver and l1 penalty norm gave the best result.

This model's different scores are shown in *table 3* and the confusion matrix is demonstrated as *figure 9*.

Table 3

	precision	recall	f-score
Class 0	1.0	1.0	1.0
Class 1	0.969	0.939	0.954
Class 2	0.911	0.929	0.920
Class 3	0.96	0.970	0.965
Total	0.960	0.960	0.960

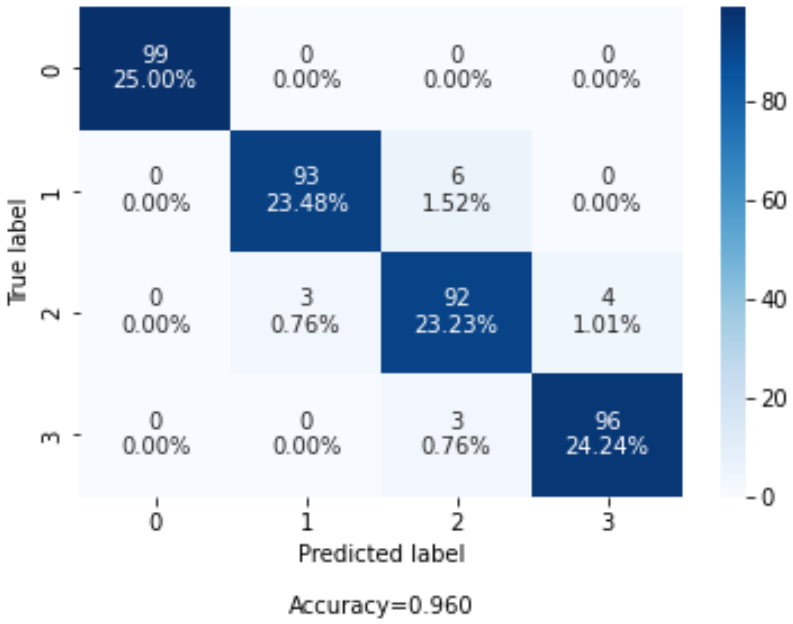

Figure 9

Since multinomial Logistic regression is by nature a multiclass classifier the terms OVO and OVA are not used to describe it.

Considering the small size of our test data, we can say that the model has probably performed the same for all classes.

PCA

Many models' performance may be affected by the number of their features. Mostly when there are a lot of features many models may perform poorly. This phenomenon is called the curse of dimensionality. Therefore, in such situations, it is necessary to drop some less important features to increase the performance of the model.

PCA is an algorithm that rearranges the space axis in a way that the data in their direction have the most possible variance. This is due to the fact that features with higher variance typically affect the model more than features with lower variances. So, by using this algorithm we may get features with higher variance compared to our original features. Also because of the nature of the PCA algorithm, we know that these features are sorted based on their variance or in other words they are probably sorted based on their effect on the mode. So if needed we can drop the last features given by the PCA knowing that they have the least variance and use the PCA algorithm as a dimensionality reduction algorithm.

While using PCA as a dimensionality reduction technique, there is a parameter called POV or percentage of variance. This parameter is a number between 0 and 1 and the lower this number is, the more features are dropped.

To see whether using PCA will improve the performance of our model, PCA with different POVs are performed on our dataset. The results are shown in *table 4*.

Table 4

	precision	recall	f-score
POV 0.99	0.957	0.957	0.957
POV 0.95	0.960	0960	0960
POV 0.9	0.957	0.957	0.957
POV 0.8	0.768	0.768	0.768
POV 0.75	0.674	0.674	0.674

And as it can be seen in *table 4*, PCA with high POV did not change our model's performance much while using lower POV caused the model's performance to drop significantly. In addition to this, considering the high performance of the model, we can say that our features are already good and we don't need to change or reduce them.

Imbalanced Data

When the number of records for each sample differs greatly, this may affect the model's performance. For example, suppose we have a binary classification problem in which 90 percent of our training samples are of class 0 and the other 10 percent are of class 1. Since 90 percent of the data is of class 0, if the model guesses that any entry is of class 0, it can get a 90 percent precision which may seem really good at first, but this model will work really bad when predicting values belonging to class 1. And if we give this model balanced test samples it will perform poorly.

There are many ways to handle imbalanced data, but this report gives an overview of only three of them.

The first way is to resample the training set. This approach is focused on making the training set balanced by either increasing the number of records for the rare classes using repletion, bootstrapping, SMOTE samples (called up-sampling or over-sampling) or removing some of the records of abundant classes (called down-sampling or under-sampling). Up-sampling is used when the quantity of data is insufficient whereas down-sampling is used when the quantity of data is sufficient.

Another way is to ensemble different resampled datasets. In this approach, we build n models that use all of the samples of the rare class and n-differing samples of the abundant class.

Also, another way to handle imbalanced data is to use the previous approach but fine-tune it by playing with the ratio between the rare and abundant classes. The best ratio heavily depends on the data and the models that are used.

In our dataset, our data is perfectly balanced and for each label, we have the same number of records, however, to illustrate how imbalanced data may affect the performance of the model, we merged the 3 higher price ranges into one class and labelled it as price range 4. Due to this, our problem is now reduced to a binary classification problem with having 3 times records of class 4 as records of class 0.

If we train our model on this imbalanced data and test it on a test dataset having the same distribution as our training dataset we get a really good result. The confusion matrix of this model is shown in *figure 10*.

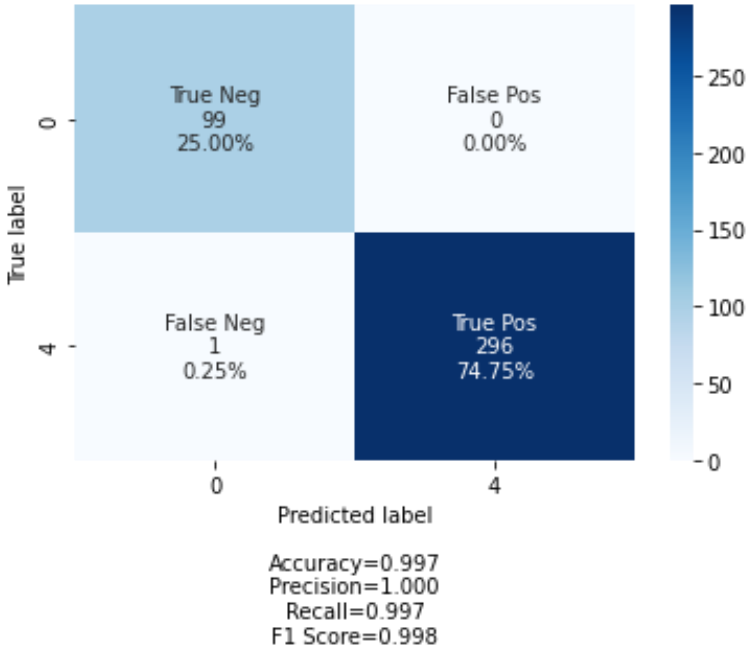

Figure 10

However, if we test the same model on a balanced test dataset the results are awful as shown in *figure 11*.

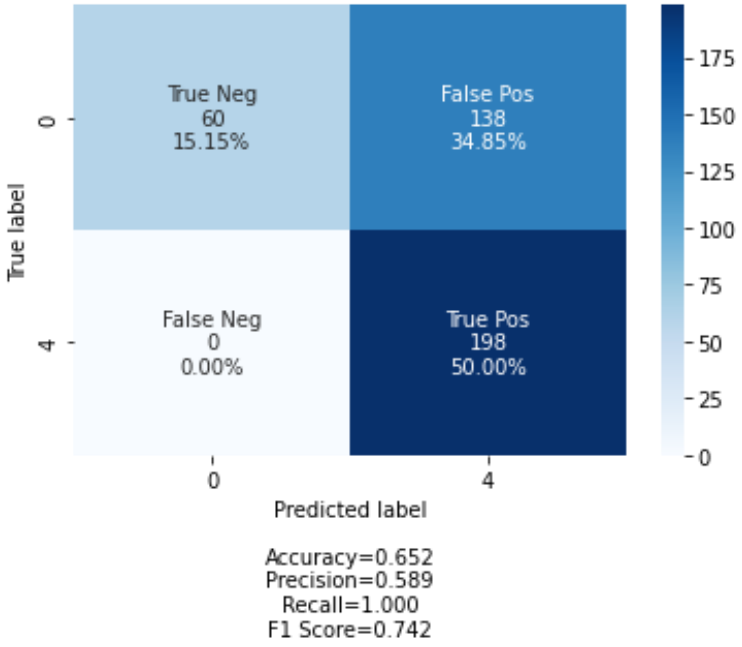

Figure 11

To solve this issue up-sampling approach was used and the model was trained on an up-sampled dataset.

The confusion matrices for testing this model on imbalanced and balanced test datasets are demonstrated in *figure 12* and *figure 13* respectively.

However, we can see that in this example, up-sampling did not solve the problem. This might be due to the fact that there are already few samples of class 0 and therefore when we take 10 percent of our total data for testing from class 0(meaning the removal of near 40 percent of class 0's records from the training set), the number of class 0's records would be too few and this may affect the performance of the model. Moreover, since the model already performs with an f-score near 0.9 on our imbalanced test data (both in binary and multinomial cases) and the

number of class 4's records is only three times the number of class 0's records, the model cannot benefit much by considering the imbalance of the dataset.

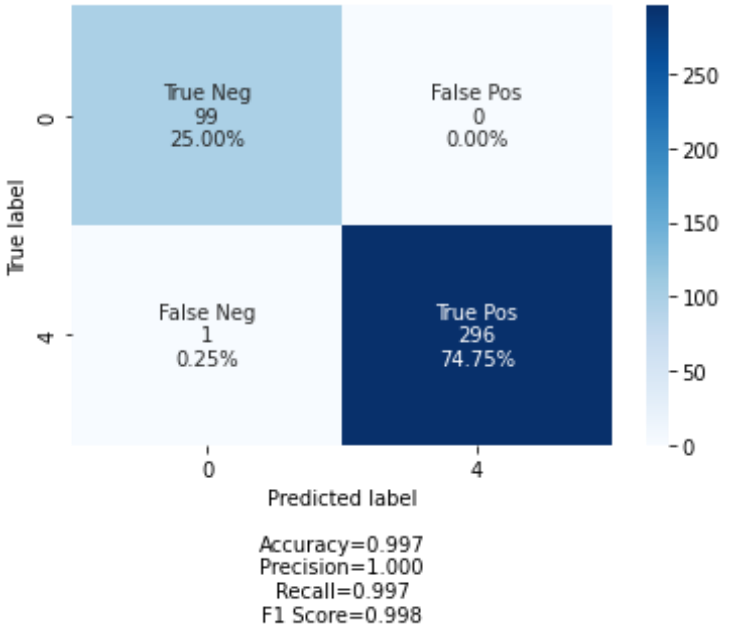

Figure 12

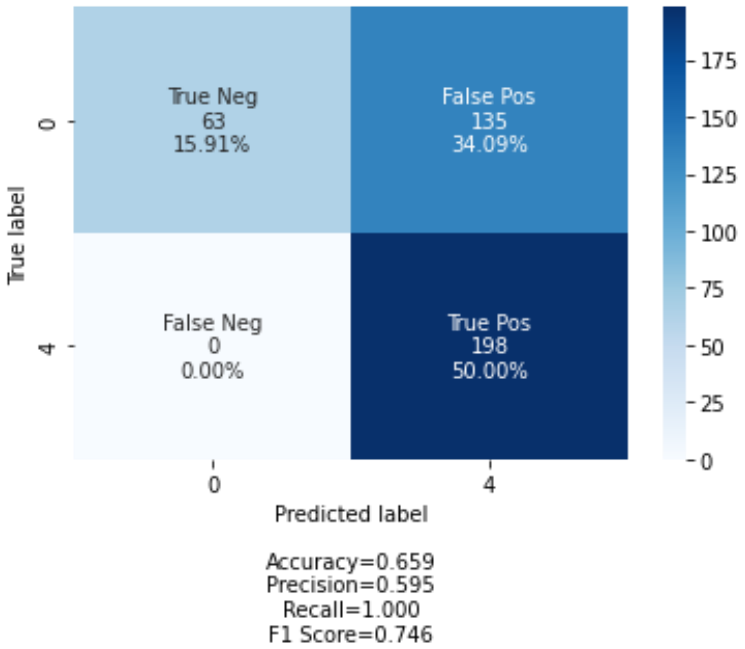

Figure 13

Although considering all of these facts, we can see in *figure 13* that the model has performed a bit better compared to *figure 11*; so we can conclude that up-sampling has made the performance on the balanced test data a bit better.

Conclusion

In the end, we can conclude that for this dataset, multinomial Logistic regression with l1 penalty norm and saga solver performs well. Also, we can conclude that it is better to standardize our data before giving it to our model.

References

https://www.analyticsvidhya.com/blog/2021/05/feature-engineering-how-to-detect-and-remove-outliers-with-python-code/

https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html

https://machinelearningmastery.com/one-vs-rest-and-one-vs-one-for-multi-class-classification/