Report: Main Project

Part 1

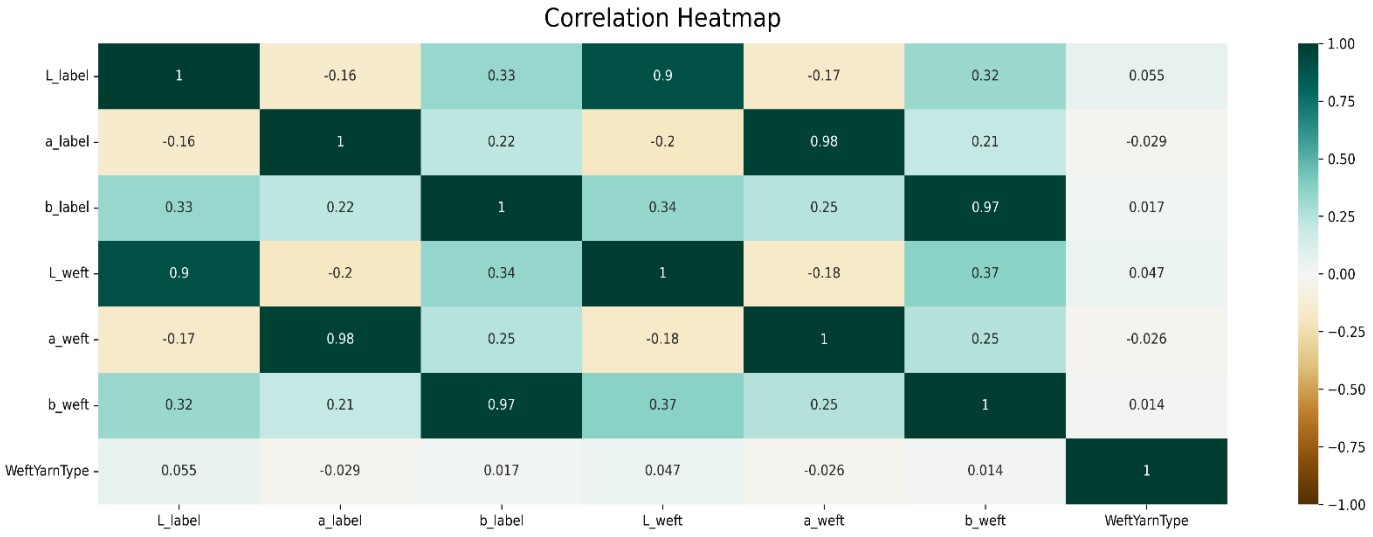
The data used in this project is from a contest in freelancer.com website that I found after doing a little bit of research. It fits the requirements of the assignment as well as it represents an excellent example for data cleaning and preprocessing before the training part. Here is the link from where I retrieved the dataset:

https://www.freelancer.com/contest/colour-prediction-2118191

The cleaning part was absolutely needed for this data, as the rows contained a big amount of Nan values (the number of rows dropped from 114947 to 59481). Also, there was a column named “Yarn Code” which will definitely not be helpful in training and testing our model. Removing none values has also an impact on our model’s score. Consequently, we need to visualize each feature and remove the none values relatedto them.

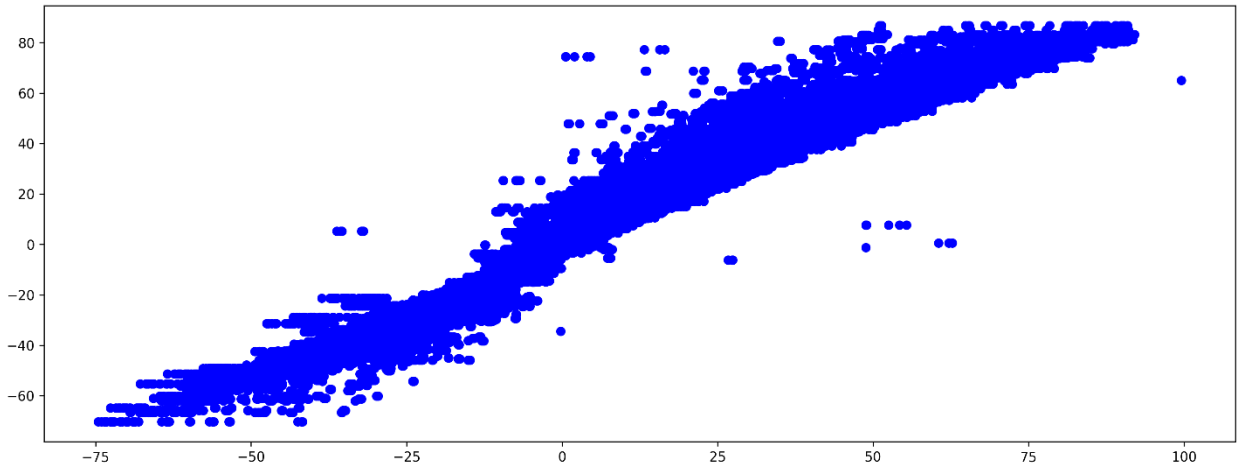
Plotting description:

Plot 1



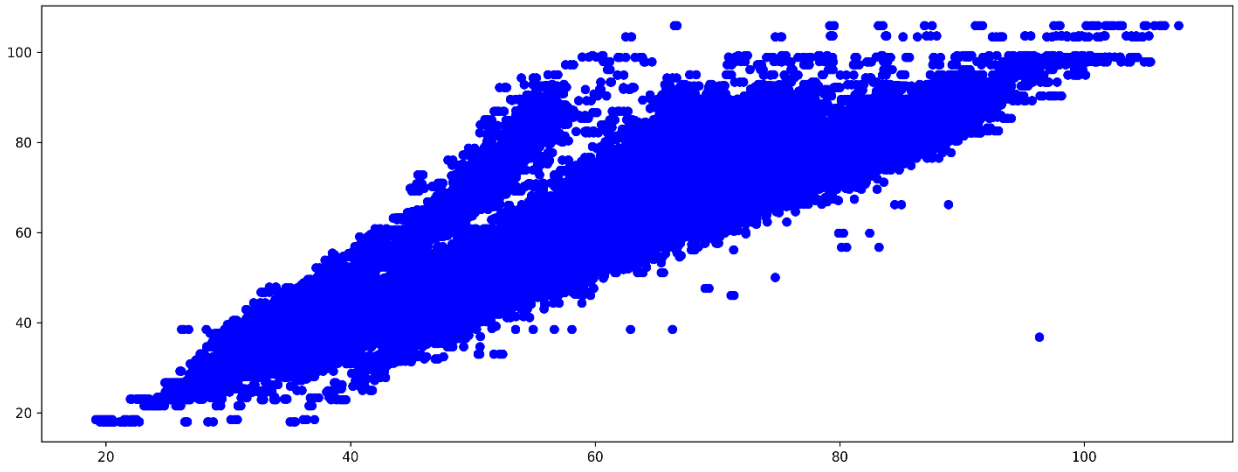
This figure (correlation matrix) displays the importance of the relationship between each feature and target. We can clearly see that there is a big correlation between ‘a_label’ and ‘a_weft’, ‘b_label’ and ‘b_weft’ and ‘L_label’ and ‘L_weft’.

Plot 2



This is a scatter plot of the values of a_weft for each value of a_label. It shows that there is a ‘linear’ relation between these two columns. Also, we can see some outliers that should be removed in order to maximize the score of our model.

Plot 3



Just like the a label, this figure is a scatter plot of the values of b_weft for each value of b_label. The relationship between these two labels is not linear, instead, the majority of the values are condensed in the center of the plot). We can visualize some outliers in this plot as well.

Part 2

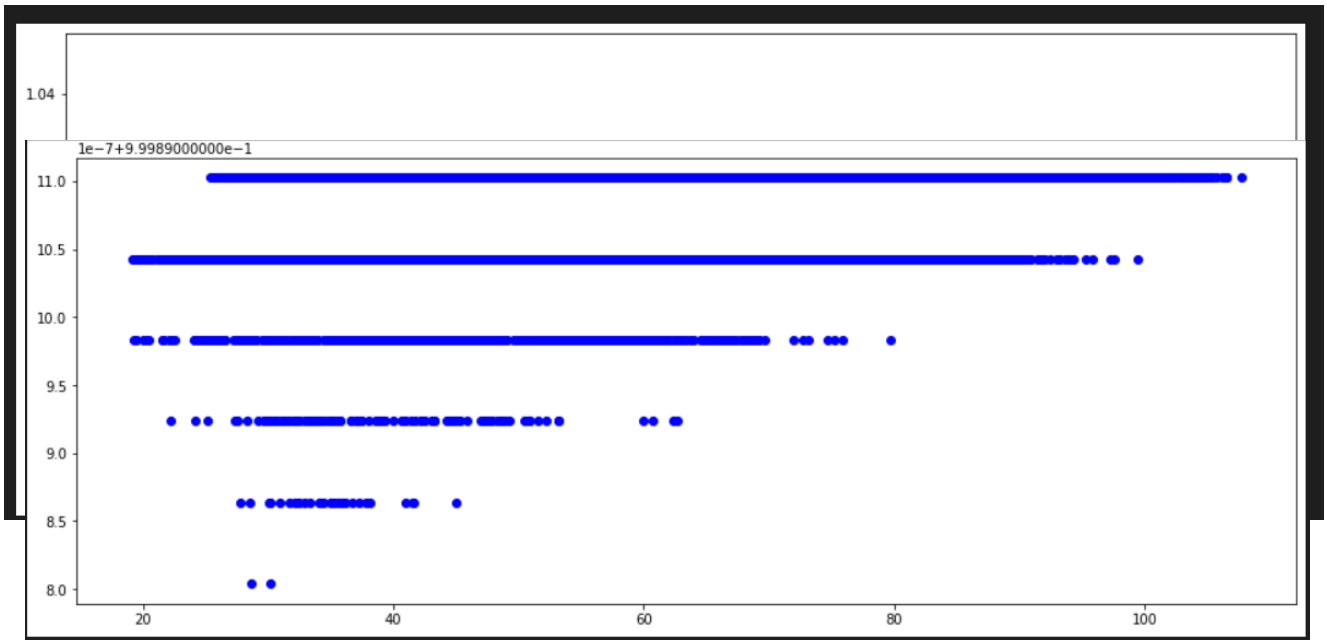
As the dataset is dedicated for regression and not classification, changes had been made in order to make our model adequate to our desires. First of all, we need drop the non-numerical values, and also drop the columns that are not needed for training the model and increasing its accuracy. Our features dataset in this stage looks like this:

	L_label	a_label	b_label	WeftYarnType
0	38.00407	36.253840	26.323170	1211
1	37.67931	36.386490	26.184560	1211
2	37.88121	36.205170	26.034750	1211
3	38.14394	37.289590	26.690500	1211
4	38.17764	37.941370	26.850100	1211
...	...	...	...	...
59476	84.92196	-5.782575	1.478362	1211
59477	86.67738	-6.509423	4.290485	1211
59478	81.18523	-5.026460	0.591886	1211
59479	81.17082	-5.086213	0.662887	1211
59480	80.92577	-5.045742	0.649619	1211

Afterwards, we convert all the numerical values into numpy.float32 numbers for accuracy purposes. Furthermore, as mentioned above, we made changes in our model such as adding some dense layers for better accuracy, changing activation functions from relu to sigmoid as sigmoid is more efficient in regression problems. We also changed the metric because accuracy metric is only for classification problems, so we decided to use mean absolute percentage error or MAPE as a regression metric and mean squared error as a loss function (mse).

Part 3:

Here we used a scatter plot to visualize how does the predictions correlate with the L_label feature. We can see that the variance of the predictions is very small, which means that the model needs to have a scaling to its variables in order to increase its accuracy. Consequently, we are going to use a min max scaler from mlxtend library, which will help us solve this problem. Here is the plot that shows the small variance of our model:



After doing all these changes, we get the following figure as the new scatter plot for our predictions related to the L_label feature:

This shows that the variance between the values is important and that the model now ‘understands’ the relation between the label and the target, also the accuracy of the model has slightly increased.

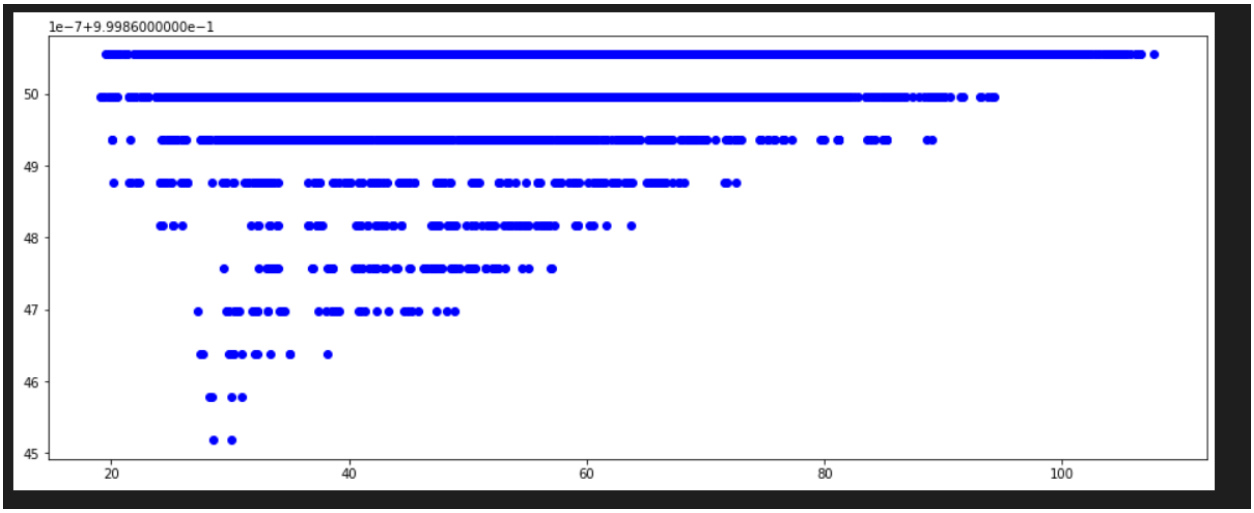
Part 4:

This part of the project is mainly focused on dealing with non-numerical values, especially that our dataset contains many text features. We will use the Label Encoder Object from Scikit-learn library. Label Encoding is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering. We will also use min max scaling as it demonstrated a big success in terms of increasing the model's accuracy. Here is how we will encode the data:

```python
#Using label encoding for all the non numerical columns
columns = ["Construction","BackingLayer2nd","PatternLayer3rd","WeftDenier","WarpColor","WeavingMachine"]

for col in columns:
    x[col] = LabelEncoder().fit_transform(x[col])


x = minmax_scaling(x,columns=[i for i in x.columns if i != "BackingLayer2nd"])

#Transforming our data into np.float32 because it's easier and better for our model
x = np.asarray(x).astype(np.float32)

x
```

0.1s Python

The label encoding helps solving the overfitting problem, because we can add more features to train on and the model now can understand the relations between the features and predict results that are more logical. In fact, the scatter plot used before has changed and gives results that are logical (L_label and predictions are getting more and more correlated).



As we can visualize here, the values are more diverse and not concentrated in only the middle of the graph, which means that the model has solved the overfitting problem.