Project Report

on

Musk Or Not Musk

Submitted by

Kavish Goyal

1. Introduction

Problem Statement:

The Ultimate goal of this project is to train a model based on the given data to classify the compounds as either 'Musk' or 'Non-Musk' compounds. The given dataset contains details about organic chemical compounds including their chemical features, isomeric conformation, names and the classes in which they are classified. The related tasks to this problem include:

- Exploratory data analysis – Understanding the type of features in the dataset, how much missing values of different features, how many outliers are existed and whether specific features are skewed, are crucial in order to create a good model in the end.
- Feature preprocessing –Preprocess the data using the insights obtained from the exploratory data analysis, potentially including feature transformation, target encoding, data type transformation, outlier detection and imputing missing values, scaling.
- Benchmark modeling – Creating a model using a standard technique for the problem in order to set up a benchmark for the future modeling improvement.

Metrics :

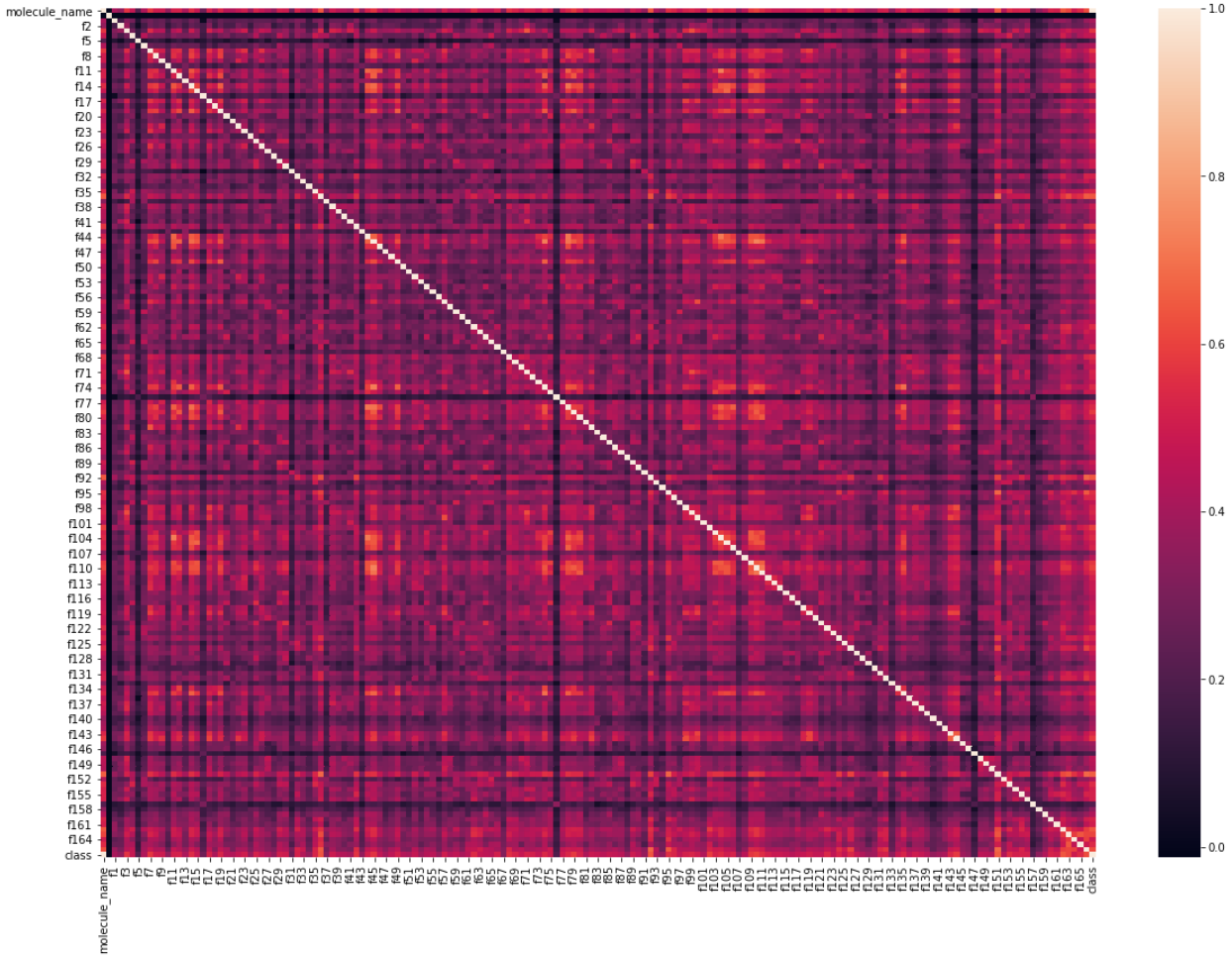
The metrics we are using for evaluation is Accuracy, Categorical-Crossentropy Loss, F1 Score, Precision, Recall.

2. Analysis

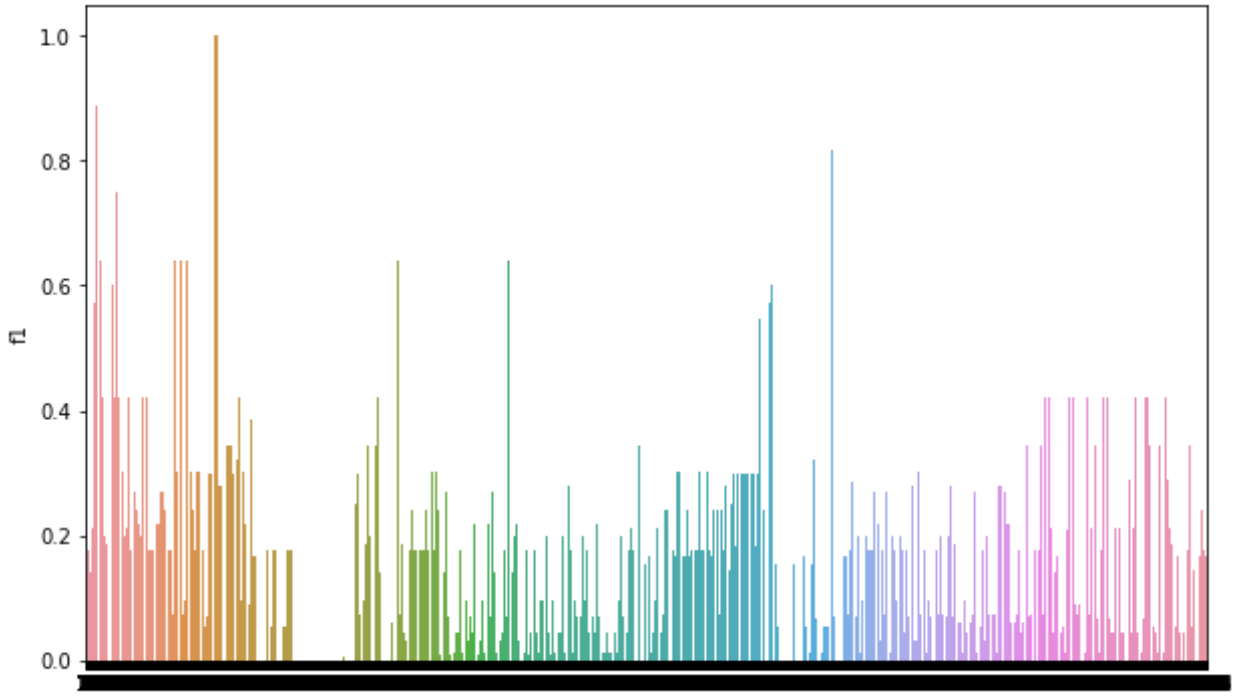
1.) Data Exploration:

The Dataset is mixed with numerical features and few categorical features. There are 170 features including the classes (musk or not-musk) of the compounds. Our task is to build a model to classify the compounds as musk or not-musk as accurately as possible.

Visualization of the data gives some useful information of the data and can find the correlations between the features.



As we can see the correlation between features. If we look closely, the some features (like f1, f5 etc.) appear relatively dark for all features. This suggests that these features play vital role (on average) in deciding whether it is 'Musk' or 'Not-Musk'!



Algorithms and Techniques

Given one of the purposes of this project is to explore Deep Learning classification techniques. I am going to use the Deep Neural Network technique aim to learn the weights layer by layer with Adam Optimizer (one of Stochastic Gradient Descent type) performs Backpropagation and Categorical crossentropy loss.

Deep Neural Network (DNN) consists at least three layers one input layer, one hidden layer and one output layer. I am using one flatten layer in the beginning to flat the data. And then the hidden layer and output layer followed by the input layer. Each layer takes its number of neurons (nodes) and activation function as its argument. I am using 'Relu' Activation in first two layers and 'Softmax' Activation in output layer.

The another Technique I am using is Multilayer perceptron (MLP) which is a class of feedforward artificial neural network (ANN).

And the last technique I am using is Support Vector Classifier(SVC) with RandomizedSearchCV for automatic hyperparameter tunning. Because SVC works better with vectors.

Data Preprocessings

- Log transform skewed features

All features that have a skewness larger than 0.75 are log transformed.

• Handle categorical features

The categorical features are handled by the category encoder’s target encoding function that converts categorical variables into dummy or indicator variables.

• Scaling and imputing

I used a pipeline for implementing standard scaling and imputing. Feature **Scaling** is a technique to standardize the independent features present in the **data** in a fixed range. It is performed to handle highly varying magnitudes or values or units. So, we use Feature **Scaling** to bring all values to same magnitudes.

Imputing is the technique to fill the missing values by its mean or median. I am using median to fill these values (here, we do not have any missing values, but I used it just for worse condition).

Implementation

At first I splitted the dataset into training and test sets with 80:20. Then trained the DNN model using training dataset. And test it on the test dataset.

As we saw that we achieve 83.33% validation accuracy.

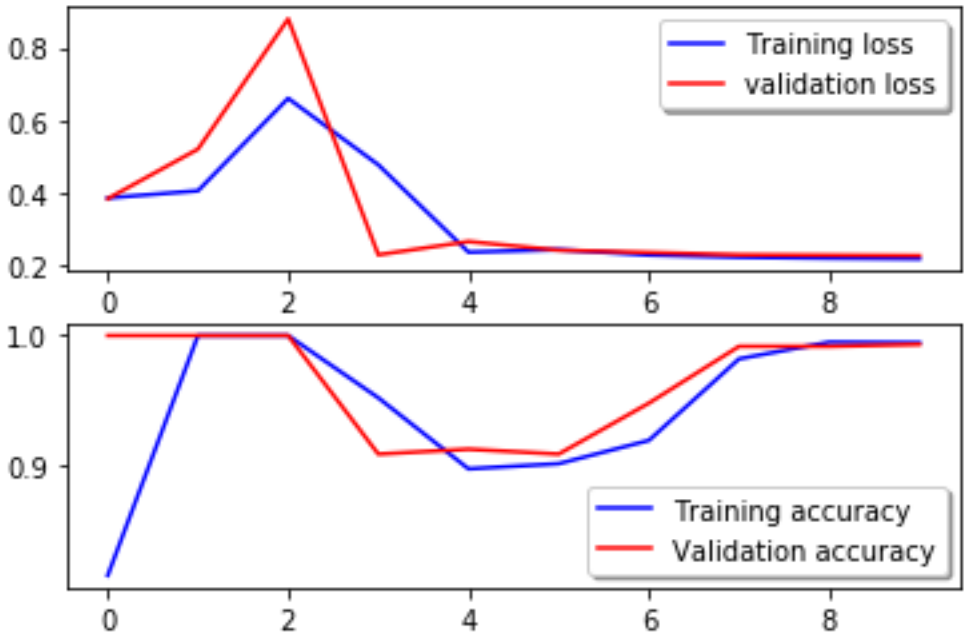
Then I trained it again with SVC and RandomizedSearchCV. Then evaluated this model on test dataset.

Here we have the confusion matrix for evaluation.

Finally, I trained the MLP on training dataset and test in on test dataset.

Results

Here is the result of DNN :



	precision	recall	f1-score	support
0	1.00	0.01	0.02	1111
1	0.16	1.00	0.27	209
accuracy			0.17	1320
macro avg	0.58	0.50	0.15	1320
weighted avg	0.87	0.17	0.06	1320

The result of SVC:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1111
1	1.00	1.00	1.00	209
accuracy			1.00	1320

macro avg	1.00	1.00	1.00	1320
weighted avg	1.00	1.00	1.00	1320

The result of MLP:

	precision	recall	f1-score	support
0	1.00	0.78	0.88	1111
1	0.46	1.00	0.63	209
accuracy			0.82	1320
macro avg	0.73	0.89	0.75	1320
weighted avg	0.91	0.82	0.84	1320

Conclusion

Our DNN and MLP model had good accuracy even on test set but low F-1 Score but there is still room for improvement.
But SVC performs best for everything as It has perfect score for all.
Here is the source link for this project(I have the source code link in the project description):
https://github.com/kvtheckrock?tab=projects

Future Work

This project opens several avenues for future work. One part is for data preprocessing, such as more advanced outlier detection techniques, more advanced feature selection techniques, fine tuning of the log transform threshold of the skewed variable and so on. More creative feature engineering will be also extremely useful. In a real world scenario, we can use such a model for binary classification.
We can try out Transfer Learning as well. For Transfer Learning we need a Pretrained model on similar task and then we can use that model for the same task by removing or adding some layers to the model as per our need. Transfer Learning can take our model's accuracy at the next level.

THANK YOU