Predicting-musk-non-musk-molecule

Deepak Jaiswal

19/02/2020

Contents

1. INTRODUCTION

1.1 Problem statement

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 compounds are classified as either 'Musk' or 'Non-Musk' compounds.

1.2 Data Set

Our task is to build a classification model on the given data using any Deep Learning approach that you deem appropriate viz. Multi-Layer Perceptron, CNN, RNN, etc.

```
data.head()
```

Out[36]:

	ID	molecule_name	conformation_name	f1	f2	f3	f4	f5	f6	f7	...	f158	f159	f160	f161	f162	f163	f164	f165	f166	class
0	1	MUSK-211	211_1+1	46	-108	-60	-69	-117	49	38	...	-308	52	-7	39	126	156	-50	-112	96	1
1	2	MUSK-211	211_1+10	41	-188	-145	22	-117	-6	57	...	-59	-2	52	103	136	169	-61	-136	79	1
2	3	MUSK-211	211_1+11	46	-194	-145	28	-117	73	57	...	-134	-154	57	143	142	165	-67	-145	39	1
3	4	MUSK-211	211_1+12	41	-188	-145	22	-117	-7	57	...	-60	-4	52	104	136	168	-60	-135	80	1
4	5	MUSK-211	211_1+13	41	-188	-145	22	-117	-7	57	...	-60	-4	52	104	137	168	-60	-135	80	1

2. Methodology

2.1 Pre Processing

Any predictive modeling requires that we look at the data before we start modeling.However, in data mining terms *looking at data* refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis.

2.1.1 Exploratory Data Analysis

In exploring the data we have

- First we Count the all Data through Command

  data.count()

- Then Describe the data for analysis.

  data.describe()

- we see number of unique values in each column of the data

  data.nunique()

- Since Id and confronation names are unique ,we can remove them.

- We take copy of data and named it new_data.

  new_data=data.copy()

- Since ID, molecule_name and conformation_name are the features which does not affect the classification and therefore can be drop to improve the accuracy of the model

2.1.2 Missing Value Analysis

Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, knn method to impute missing value.

In python - **new_data.isnull().values.any()** is used to detect any missing value

There is no missing values found in given Dataset

2.1.3 Splitting data into training and test dataset into inputs and targets in ratio 80:20

We need to split up our dataset into inputs (train_X) and our target (train_y).

Importing and using Keras model because Keras is an Open Source Neural Network library written in Python that runs on top of Theano or Tensorflow.

It is designed to be modular, fast and easy to use.Keras doesn't handle low-level computation. Instead, it uses another library to do it, called the "Backend.

So Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Keras High-Level API handles the way we make models, defining layers, or set up multiple input-output models. In this level, Keras also compiles our model with loss and optimizer functions, training process with fit function.

Training the Model

Now we will train our model. To train, we will use the 'fit()' function on our model with the following five parameters: training data (train_X), target data (train_y), validation split, the number of epochs and callbacks.

The validation split will randomly split the data into use for training and testing. During training, we will be able to see the validation loss, which give the mean squared error of our model on the validation set. We will set the validation split at 0.2, which means that 20% of the training data we provide in the model will be set aside for testing model performance.

Compiling the model

Next, we need to compile our model. Compiling the model takes two parameters: optimizer and loss.

The optimizer controls the learning rate. We will be using 'adam' as our optmizer. Adam is generally a good optimizer to use for many cases. The adam optimizer adjusts the learning rate throughout training.

The learning rate determines how fast the optimal weights for the model are calculated. A smaller learning rate may lead to more accurate weights (up to a certain point), but the time it takes to compute the weights will be longer.

For our loss function, we will use ‘mean_squared_error’. It is calculated by taking the average squared difference between the predicted and actual values. It is a popular loss function for regression problems. The closer to 0 this is, the better the model performed.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

where N is the number of data points,
f_i the value returned by the model and
y_i the actual value for data point i.

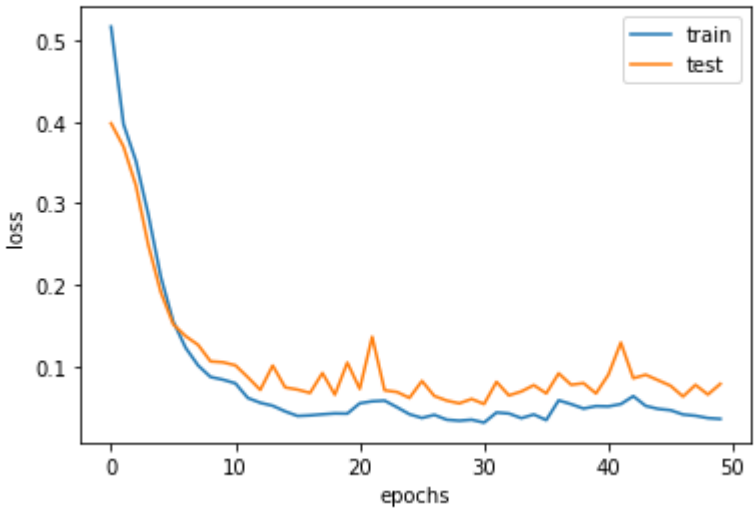
```
Use tf.cast instead.
Train on 5278 samples, validate on 1320 samples
Epoch 1/50
 - 2s - loss: 0.5165 - accuracy: 0.7444 - val_loss: 0.3981 - val_accuracy: 0.8485
Epoch 2/50
 - 0s - loss: 0.3971 - accuracy: 0.8452 - val_loss: 0.3695 - val_accuracy: 0.8485
Epoch 3/50
 - 0s - loss: 0.3519 - accuracy: 0.8452 - val_loss: 0.3219 - val_accuracy: 0.8492
Epoch 4/50
 - 0s - loss: 0.2859 - accuracy: 0.8820 - val_loss: 0.2486 - val_accuracy: 0.9189
Epoch 5/50
 - 0s - loss: 0.2100 - accuracy: 0.9299 - val_loss: 0.1910 - val_accuracy: 0.9364
Epoch 6/50
 - 0s - loss: 0.1552 - accuracy: 0.9490 - val_loss: 0.1518 - val_accuracy: 0.9447
Epoch 7/50
 - 0s - loss: 0.1229 - accuracy: 0.9591 - val_loss: 0.1372 - val_accuracy: 0.9477
Epoch 8/50
 - 0s - loss: 0.1013 - accuracy: 0.9655 - val_loss: 0.1265 - val_accuracy: 0.9568
Epoch 9/50
 - 0s - loss: 0.0872 - accuracy: 0.9699 - val_loss: 0.1063 - val_accuracy: 0.9652
Epoch 10/50
 - 0s - loss: 0.0839 - accuracy: 0.9714 - val_loss: 0.1052 - val_accuracy: 0.9583
Epoch 11/50
 - 0s - loss: 0.0794 - accuracy: 0.9701 - val_loss: 0.1015 - val_accuracy: 0.9636
Epoch 12/50
 - 0s - loss: 0.0615 - accuracy: 0.9765 - val_loss: 0.0869 - val_accuracy: 0.9689
Epoch 13/50
 - 0s - loss: 0.0556 - accuracy: 0.9805 - val_loss: 0.0714 - val_accuracy: 0.9742
Epoch 14/50
 - 0s - loss: 0.0519 - accuracy: 0.9820 - val_loss: 0.1012 - val_accuracy: 0.9621
Epoch 15/50
 - 0s - loss: 0.0450 - accuracy: 0.9839 - val_loss: 0.0745 - val_accuracy: 0.9742
```

Recurrent Convolutional Neural Networks (RCNN)

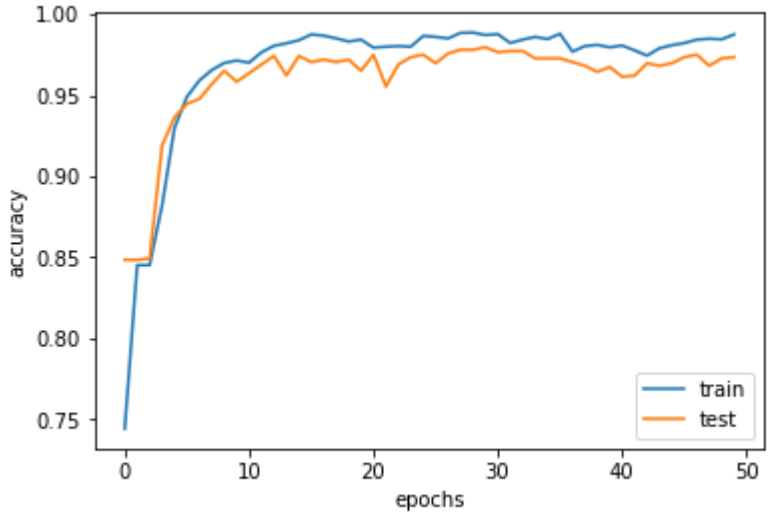
Recurrent Convolutional Neural Networks (RCNN) is also used for text classification. The main idea of this technique is capturing contextual information with the recurrent structure and constructing the representation of text using a convolutional neural network. This architecture is a combination of RNN and CNN to use the advantages of both technique in a model.

Loss curve for training and test Data Set

```
plt.legend()
plt.show()
```



Accuracy curve for training and test dataset



Final performance measures of our model including validation accuracy, loss, precision, recall, F1 score

##Final performance measures of our model including
loss, precision, recall, F1 score.

29]:
```
from sklearn.metrics import classification_report
cls = classification_report(Y_test,Y_pred)
print(cls)
```

```
              precision    recall  f1-score   support

           0       0.85      1.00      0.92      1120
           1       0.00      0.00      0.00       200

    accuracy                           0.85      1320
   macro avg       0.42      0.50      0.46      1320
weighted avg       0.72      0.85      0.78      1320
```

30]:
```
model.save('C:\\Users\\Deepak Jaiswal\\Desktop\\Credicxo _Assignment\\model2.h5')
```

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

Approaches and Solutions

Given that the target variable activity is continuous and each dataset contains tens of thousands of entries, this is a regression problem and I will build a neural network(NN) using the TensorFlow library in Python to make predictions. The cost function is defined as mean squared error or MSE in this work