A Report on classification of Compounds as Musk or Non-Musk.

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.

In this project, I build a classification model using multi-layer perceptron in python. I use online notebook i.e. google colab.

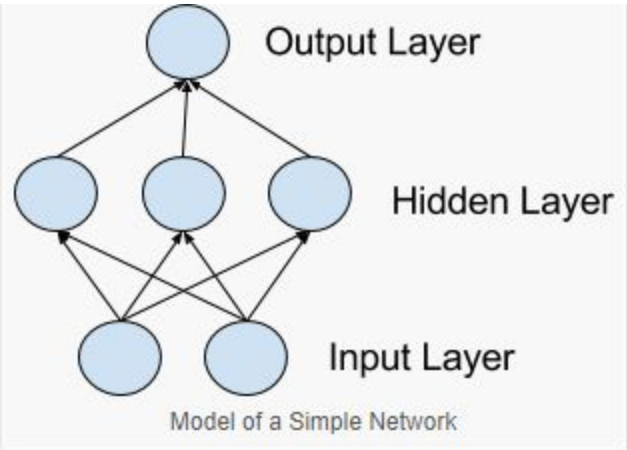
Python :- Python is a high-level, interpreted, interactive and object-oriented scripting language. It is easy to learn and use. It comprises of large standard library. It is used for data analysis,machine learning,data visualization.

There are many python packages used in this project. Some are:-

- Numpy :- Fundamental package for scientific computing and useful for linear algebra.
- Pandas :- open source and high performance easy to use data structure and data analysis tools used for data manipulation.
- Matplotlib :- Python 2D plotting library.
- Sklearn:- Simple and efficient tools for data mining and data analysis.

Multilayer Perceptron(MLP) :- It is classical type of neural network.They are comprised of one or more layers of neurons. Data is fed to the input layer, there may be one or more hidden layers providing levels of abstraction, and predictions are made on the output layer, also called the visible layer.

MLP are suitable for regression problems, classification problems and data is provided into tabular forms.



In this project, data is provided into tabular form i.e. csv formats and it is basically binary classification problem. For that, MLP is very flexible and can be used to learn a mapping from inputs to outputs. So, i use MLP to solve this problem.

SOURCE CODE AND OUTPUT

I load data using:-

```
data=pd.read_csv("/content/drive/My Drive/data/musk_csv.csv")
```

```
data=pd.read_csv("/content/drive/My Drive/data/musk_csv.csv")

[276] data
```

	ID	molecule_name	conformation_name	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	f11	f12	f13	f14	f15	f16	f17
0	1	MUSK-211	211_1+1	46	-108	-60	-69	-117	49	38	-161	-8	5	-323	-220	-113	-299	-283	-307	-31
1	2	MUSK-211	211_1+10	41	-188	-145	22	-117	-6	57	-171	-39	-100	-319	-111	-228	-281	-281	-300	54
2	3	MUSK-211	211_1+11	46	-194	-145	28	-117	73	57	-168	-39	-22	-319	-111	-104	-283	-282	-303	52
3	4	MUSK-211	211_1+12	41	-188	-145	22	-117	-7	57	-170	-39	-99	-319	-111	-228	-282	-281	-301	54
4	5	MUSK-211	211_1+13	41	-188	-145	22	-117	-7	57	-170	-39	-99	-319	-111	-228	-282	-281	-301	54
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...

In this project, after loading the data, I checked description of data using “data.describe()” and find number of values , minimum, maximum , mean values e.t.c data and I checked that there is no missing values and there is no null values using “data.isnull().sum().any()”. I delete columns like Id, molecular_name, conformation_name as these are irrelevant data for our models. I split the input

data into training and test data using “train_test_split”. for training data using multilayer perceptron,

USING MULTI-LAYER PERCEPTRON(MLP)

```
[288] import tensorflow as tf
      import keras
      from keras.models import Sequential
      from keras.layers import Dense

[289] classifier=Sequential()

[290] classifier.add(Dense(units=15,kernel_initializer='uniform',activation='relu',input_dim=166))

[291] classifier.add(Dense(units=15,kernel_initializer='uniform',activation='relu'))

[292] classifier.add(Dense(units=1,kernel_initializer='uniform',activation='sigmoid'))

[293] classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

[294] history=classifier.fit(x_train,y_train,validation_split=0.2,batch_size=32,epochs=9)
```

```
Train on 4222 samples, validate on 1056 samples
Epoch 1/9
4222/4222 [==============================] - 1s 218us/step - loss: 0.3136 - acc: 0.8790 - val_loss: 0.2076 - val_acc: 0.9252
Epoch 2/9
4222/4222 [==============================] - 0s 55us/step - loss: 0.1800 - acc: 0.9330 - val_loss: 0.1788 - val_acc: 0.9347
Epoch 3/9
4222/4222 [==============================] - 0s 57us/step - loss: 0.1338 - acc: 0.9481 - val_loss: 0.1496 - val_acc: 0.9413
Epoch 4/9
4222/4222 [==============================] - 0s 55us/step - loss: 0.1150 - acc: 0.9562 - val_loss: 0.1160 - val_acc: 0.9527
Epoch 5/9
4222/4222 [==============================] - 0s 55us/step - loss: 0.0962 - acc: 0.9623 - val_loss: 0.1231 - val_acc: 0.9527
Epoch 6/9
4222/4222 [==============================] - 0s 58us/step - loss: 0.0904 - acc: 0.9671 - val_loss: 0.1182 - val_acc: 0.9564
Epoch 7/9
4222/4222 [==============================] - 0s 56us/step - loss: 0.0755 - acc: 0.9694 - val_loss: 0.0877 - val_acc: 0.9678
Epoch 8/9
4222/4222 [==============================] - 0s 55us/step - loss: 0.0723 - acc: 0.9721 - val_loss: 0.0902 - val_acc: 0.9735
Epoch 9/9
4222/4222 [==============================] - 0s 56us/step - loss: 0.0726 - acc: 0.9713 - val_loss: 0.0928 - val_acc: 0.9669
```

After training data, I got accuracy score to be `0.9734848484848485` and F1 score to be `0.911392405063291`

```
print(accuracy_score(y_test,y_pred))
0.9734848484848485

[ ] print(f1_score(y_test,y_pred))
0.9113924050632912
```

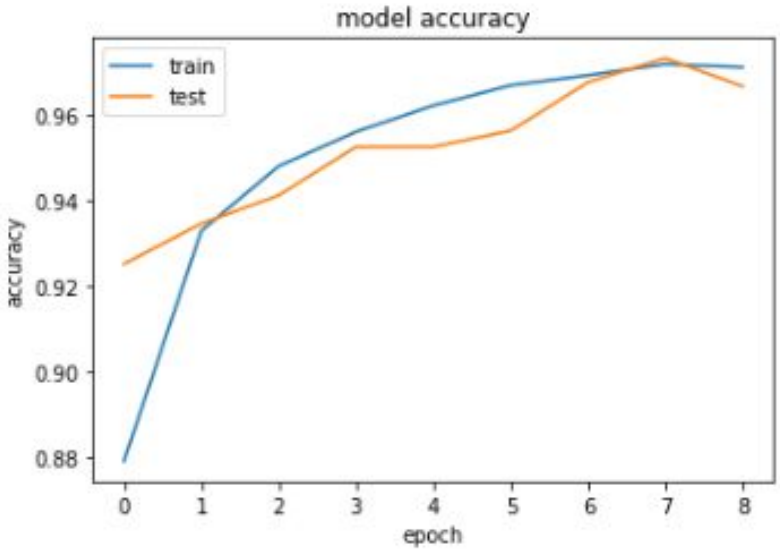
Precision and recall are respectively:-

```
[ ] print(classification_report(y_test,y_pred))
```

	precision	recall	f1-score	support
0	0.98	0.99	0.98	1117
1	0.94	0.89	0.91	203
accuracy			0.97	1320
macro avg	0.96	0.94	0.95	1320
weighted avg	0.97	0.97	0.97	1320

Accuracy graph

```
# summarize history for accuracy
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Loss graph

```
[ ]  # summarize history for loss
     plt.plot(history.history['loss'])
     plt.plot(history.history['val_loss'])
     plt.title('model loss')
     plt.ylabel('loss')
     plt.xlabel('epoch')
     plt.legend(['train', 'test'], loc='upper left')
     plt.show()
```