Brief summary

1.Label

- Manually tag each waveform
- To calculate Easily, label value 0-4 corresponds to 140-148

2. Import Data

- $X : 100(num) * 302(factor)$
- $Y : 100 * 1$
- We can choose to use PCA for the number of factor is large

```
dframe = pd.read_excel("test.xlsx")
test = DataFrame(dframe)
test=test.T
X = test.iloc[ : , :-1].values
#X = finalData  # Using PCA compressed data, the test results are not ideal
Y = test.iloc[ : , 302].values
```

3. Data Preprocessing

1. For Multi-layer Neural Network

- First we neeed to split the dataset : Using Sklearn split automatically(or using "k-fold cross validation")
- Change y to One-hot vector like $0->[1,0,0,0,0];1->[0,1,0,0,0];$

```
# Using sklearn to split dataset 1:4
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.20, random_state=0)

# shape each y to [0,0,1,0,0]....
y_train = np_utils.to_categorical(y_train)
y_test = np_utils.to_categorical(y_test)
num_classes = y_test.shape[1]
```

2. For other Classifiers

> Random tree, random forest, SVM...

- Split the dataset,type meets requirements.
- Set Random_state as 0 ; it is like random seed which needs to set before optimizing

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y,
test_size=0.20, random_state=0)
```

4. Training model

1. Multi-layer Neural Network

- Construct a three-layer neural network based on keras .
- Choosing 'relu' as the activation function
- Using Adam to optimize
- Adding validation data (20%) to reduce overfitting
- According to test, wo choose epochs as 300

```
# 3-layers
model = Sequential()
model.add(Dense(90, activation='relu', input_dim=302))
model.add(Dense(30, activation='relu'))
model.add(Dense(5, activation='softmax'))

# using Adam to optimize
rms = keras.optimizers.Adam(lr=0.0005, beta_1=0.9, beta_2=0.999, epsilon=1e-08)
model.compile(loss='binary_crossentropy',
              optimizer=rms,
              metrics=['accuracy'])
# add validation data to reduce overfitting
model.fit(x_train, y_train,validation_split=0.2,
          epochs=300,
          batch_size=10)
score = model.evaluate(x_test, y_test, batch_size=10)
print(model.metrics_names)
print(score)
```

Results: Accuracy is 98.00%

> We can continue to optimize and improve accuracy.

2. Other Classifiers

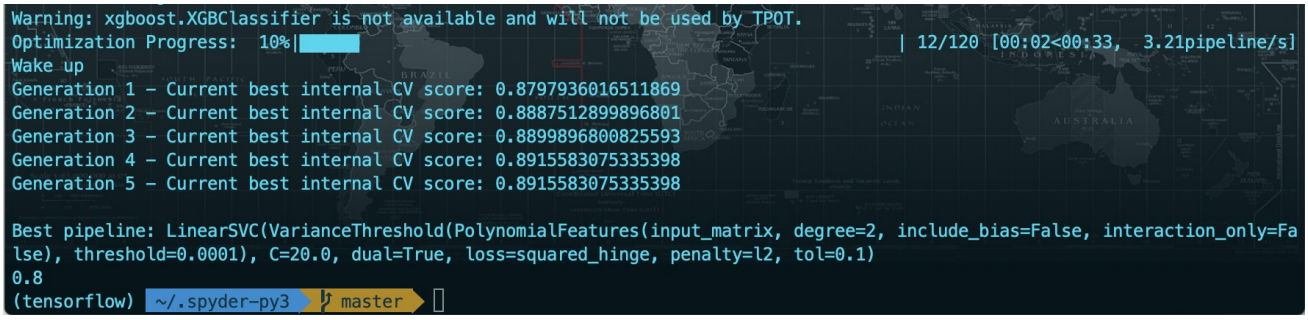
- Using tpot package to select models and optimize automatically

  > A Python Automated Machine Learning tool that optimizes machine learning pipelines using genetic programming.

```
tpot = TPOTClassifier(generations=5, population_size=20, verbosity=2)
tpot.fit(X_train, y_train)
```

```
print(tpot.score(X_test, y_test))
tpot.export('tpot_pipeline.py')
```

- Results: Accuracy is 89.15% --- Linear SVC



- Once TPOT is finished searching , it provides us with the Python code for the best pipeline it found so we can tinker with the pipeline from there.
- According to this , using SVC(linear) + PCA

```
from sklearn.svm import SVC
clf = SVC(C=1,kernel='linear', probability = True,random_state=0)
clf.fit(x_train, y_train)
```

Results : Accuracy is 90%

Reason :

- The number of features is much larger than the number of samples
- The number of samples is small .

> We can continue to optimize and improve accuracy.

5. Conclution

- Better to Use Multi-layer Neural Network
- Better to have more data
- PCA is not useful for this data , maybe useful for a larger factors
- Maybe we can use DP Alg to compressed data (waveform)

2018-11-10