Assignment 2 Report (Performance Metrics): Group 32

Overview

Problem: Develop an online Application Service that accepts Human Pose Skeletal key points of a sign video and return the label of the sign as a JSON Response. The Key points are generated using TensorFlow's Pose Net.

1.) Data Processing Phase: Collected X and Y co-ordinated from given PoseNet Files.

Features Reflected:

- Fast Fourier Transform
- Standard Deviation
- Mean

# The resultant feature vector does contain the above mentioned features.

2.) Training Phase:

Models were directly split the given dataset (features extracted from the Posenet) into training and validation sets in the ratio of 80:20. Then all the models were trained using 80% training data. Before this, pre-processing was done to extract the features.

Models Adopted:

- SupportVectorMachine
- Logistic Regression
- Linear Discriminant Analysis
- Multilayer Perceptron

# The Trained Models were stored in the pickle file as mentioned in the task space.

#Service URL: http://g32-gesture-prediction.herokuapp.com/

3.) Back-End Design Structure:

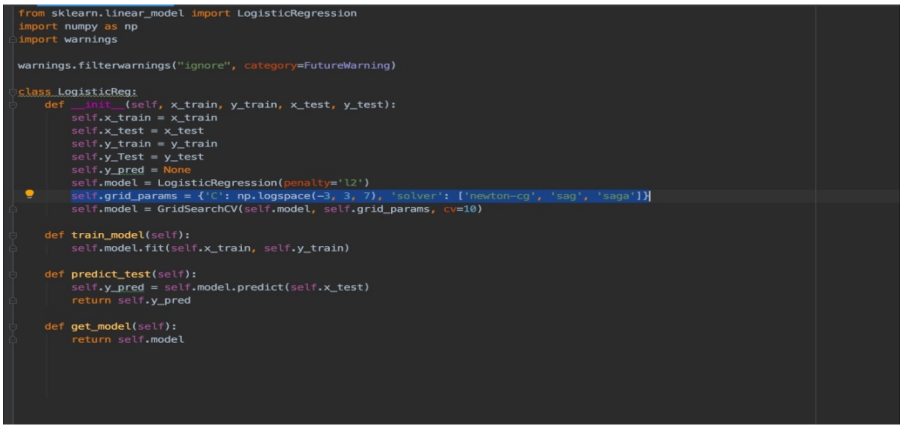
Then equipped with Flask (Powerful Python web Framework) we enabled RESTful services for our Data sets and ML models, Flask also does have the internal support for http "get" and "post" mechanisms. Then we take into account the JSN file and store it into a "NumPy" Array.

There after using the generated pickle file to extract trained models hence applying those to test the "Given Data Set". Finally we produced JSN output predictions for out ML Models incorporated with the data set.

MODEL DESCRIPTION AND ASSOCIATED ACCURACIES:

Logistic Regression:

Parameters Used: C, Solver

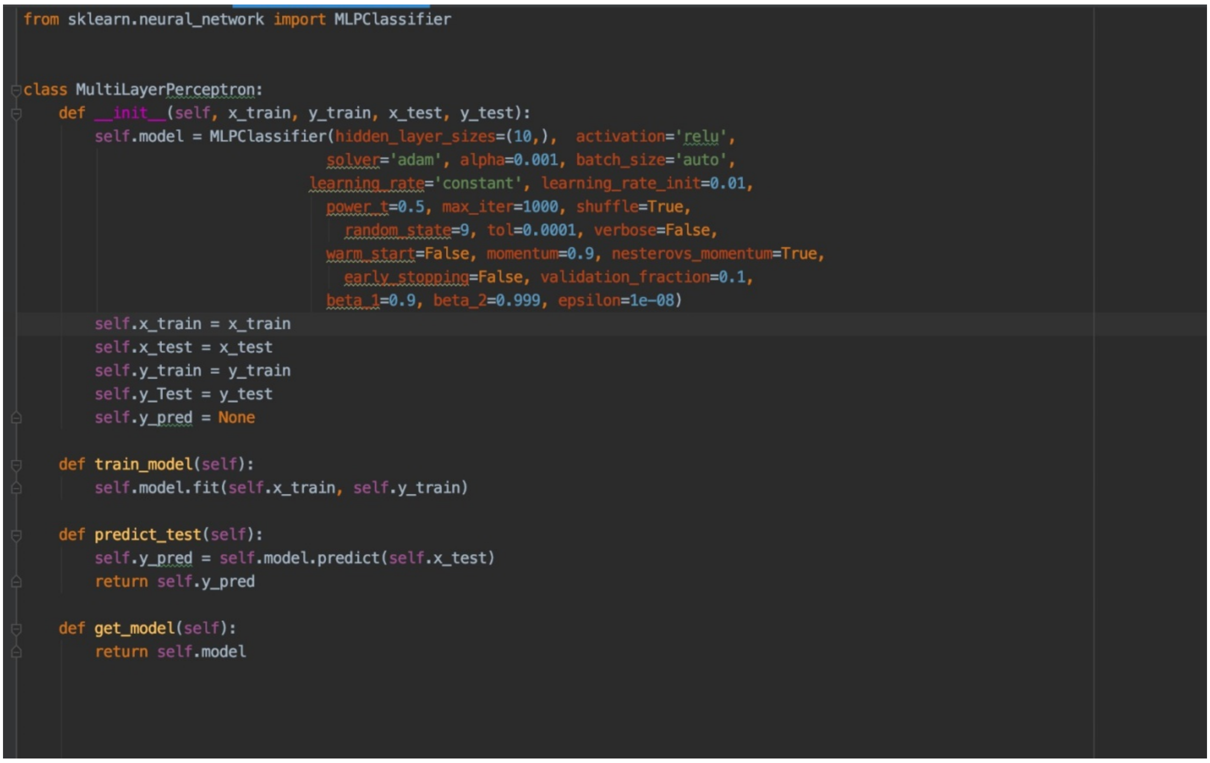


```
from sklearn.linear_model import LogisticRegression
import numpy as np
import warnings

warnings.filterwarnings("ignore", category=FutureWarning)

class LogisticReg:
    def __init__(self, x_train, y_train, x_test, y_test):
        self.x_train = x_train
        self.x_test = x_test
        self.y_train = y_train
        self.y_Test = y_test
        self.y_pred = None
        self.model = LogisticRegression(penalty='l2')
        self.grid_params = {'C': np.logspace(-3, 3, 7), 'solver': ['newton-cg', 'sag', 'saga']}
        self.model = GridSearchCV(self.model, self.grid_params, cv=10)

    def train_model(self):
        self.model.fit(self.x_train, self.y_train)

    def predict_test(self):
        self.y_pred = self.model.predict(self.x_test)
        return self.y_pred

    def get_model(self):
        return self.model
```

Accuracy: 78.31%

Multi-Layer Perceptron:

Parameters Used : activation, solver, alpha, batch_size, learning_rate, power_t, max_item, shuffle, random_state, tol, verbose, warm_start=False, momentum=0.9, nesterovs_momentum, early_stopping, validation_fraction,beta_1, beta_2, epsilon.

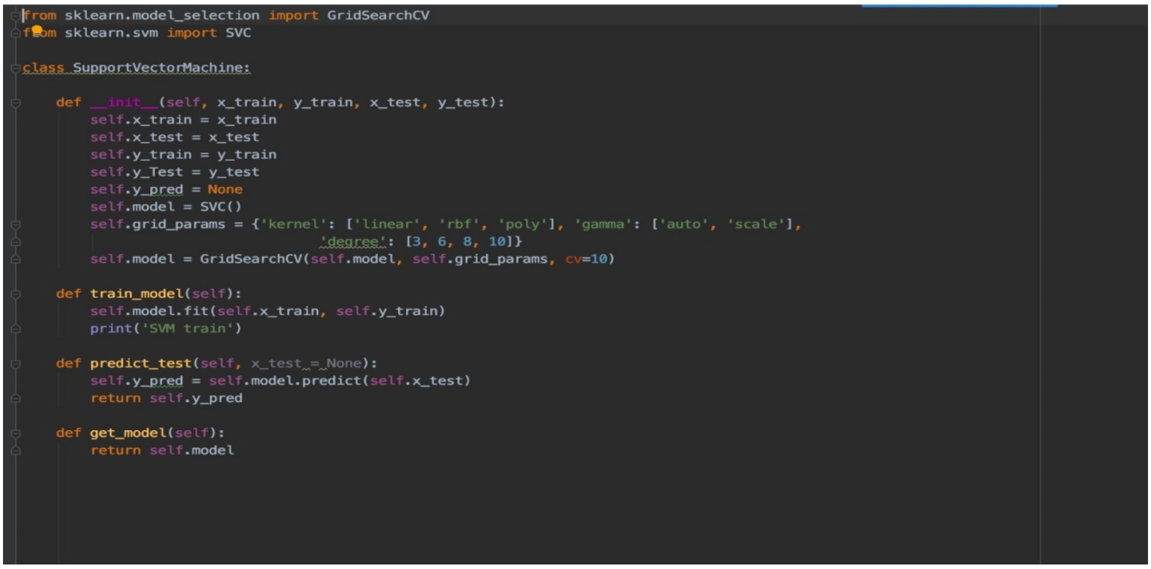


```
from sklearn.neural_network import MLPClassifier


class MultiLayerPerceptron:
    def __init__(self, x_train, y_train, x_test, y_test):
        self.model = MLPClassifier(hidden_layer_sizes=(10,),  activation='relu',
                                  solver='adam', alpha=0.001, batch_size='auto',
                                learning_rate='constant', learning_rate_init=0.01,
                                  power_t=0.5, max_iter=1000, shuffle=True,
                                    random_state=9, tol=0.0001, verbose=False,
                                  warm_start=False, momentum=0.9, nesterovs_momentum=True,
                                    early_stopping=False, validation_fraction=0.1,
                                  beta_1=0.9, beta_2=0.999, epsilon=1e-08)
        self.x_train = x_train
        self.x_test = x_test
        self.y_train = y_train
        self.y_Test = y_test
        self.y_pred = None

    def train_model(self):
        self.model.fit(self.x_train, self.y_train)

    def predict_test(self):
        self.y_pred = self.model.predict(self.x_test)
        return self.y_pred

    def get_model(self):
        return self.model
```

Accuracy: 73.49%

Support Vector Machines:

Parameters Used: Kernel, gamma, degree

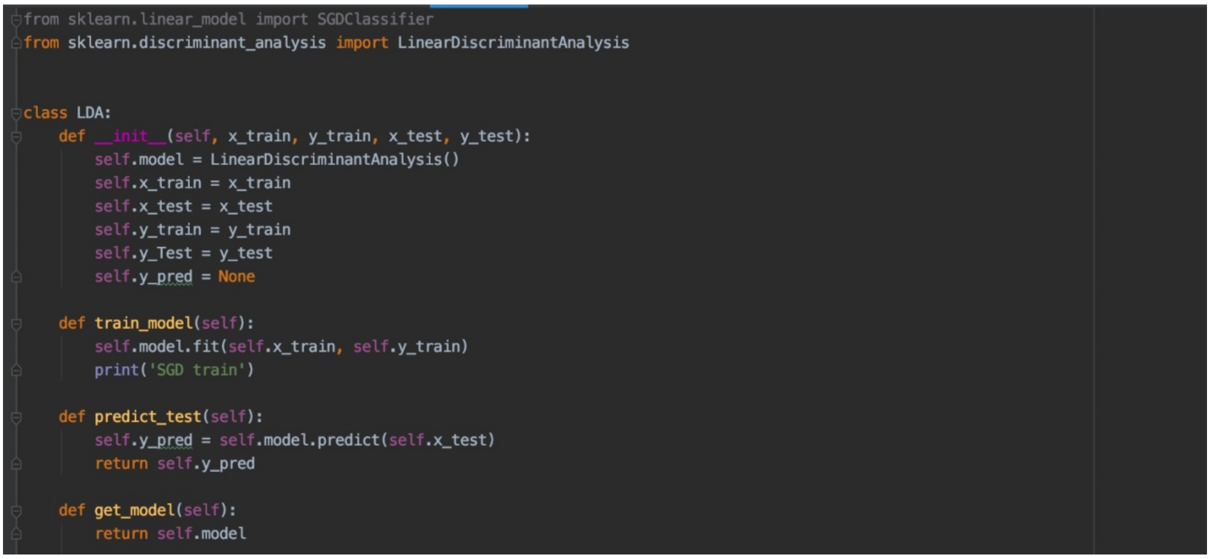


```python
from sklearn.model_selection import GridSearchCV
from sklearn.svm import SVC

class SupportVectorMachine:

    def __init__(self, x_train, y_train, x_test, y_test):
        self.x_train = x_train
        self.x_test = x_test
        self.y_train = y_train
        self.y_Test = y_test
        self.y_pred = None
        self.model = SVC()
        self.grid_params = {'kernel': ['linear', 'rbf', 'poly'], 'gamma': ['auto', 'scale'],
                           'degree': [3, 6, 8, 10]}
        self.model = GridSearchCV(self.model, self.grid_params, cv=10)

    def train_model(self):
        self.model.fit(self.x_train, self.y_train)
        print('SVM train')

    def predict_test(self, x_test = None):
        self.y_pred = self.model.predict(self.x_test)
        return self.y_pred

    def get_model(self):
        return self.model
```

Accuracy: 81.92%

Linear Discriminant Analysis:

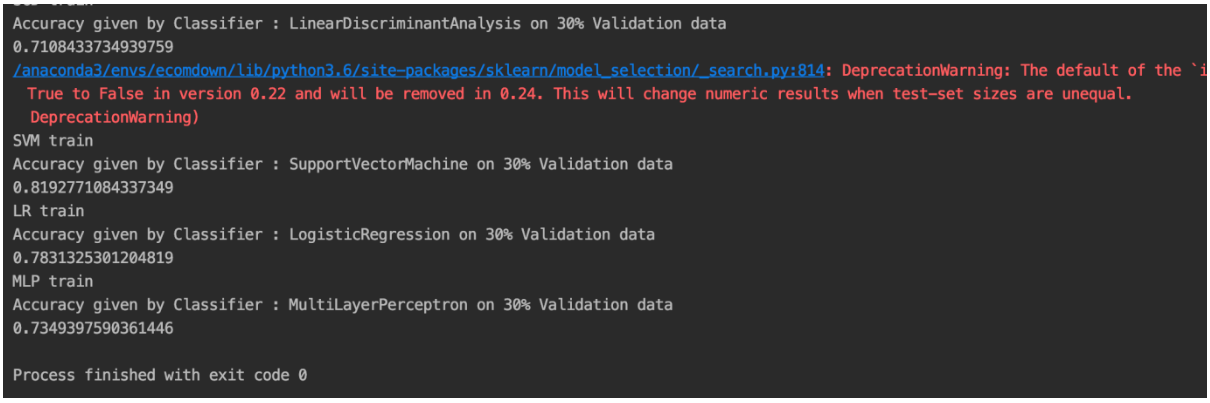
Parameters Used: solver, shrinkage, priors, n_components, store_covariance, tol



```python
from sklearn.linear_model import SGDClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis


class LDA:
    def __init__(self, x_train, y_train, x_test, y_test):
        self.model = LinearDiscriminantAnalysis()
        self.x_train = x_train
        self.x_test = x_test
        self.y_train = y_train
        self.y_Test = y_test
        self.y_pred = None

    def train_model(self):
        self.model.fit(self.x_train, self.y_train)
        print('SGD train')

    def predict_test(self):
        self.y_pred = self.model.predict(self.x_test)
        return self.y_pred

    def get_model(self):
        return self.model
```

Accuracy: 71.08%

Accuracy Evidence:



```
Accuracy given by Classifier : LinearDiscriminantAnalysis on 30% Validation data
0.7108433734939759
/anaconda3/envs/ecomdown/lib/python3.6/site-packages/sklearn/model_selection/_search.py:814: DeprecationWarning: The default of the `i
  True to False in version 0.22 and will be removed in 0.24. This will change numeric results when test-set sizes are unequal.
  DeprecationWarning)
SVM train
Accuracy given by Classifier : SupportVectorMachine on 30% Validation data
0.8192771084337349
LR train
Accuracy given by Classifier : LogisticRegression on 30% Validation data
0.7831325301204819
MLP train
Accuracy given by Classifier : MultiLayerPerceptron on 30% Validation data
0.7349397590361446

Process finished with exit code 0
```

Contributions and Credits:

SupportVectorMachine - Krishna Chaitanya Bogavalli

Logistic Regression - Itish

LinearDiscriminantAnalysis - Vinisha Sukameti

MultilayerPerceptron - Prasanth Reddy