SIGN LANGUAGE RECOGNITION

Term Paper (INT417 - MACHINE LEARNING ALGORITHMS)

Maddula Vishnu Vardhan Reddy – 11909034
Programme- P132: B.Tech
(Department of Computer Sciences and Engineering)
Lovely Professional University, Punjab

ABSTRACT:

Inability to speak is true disability and people with this disability use different modes to communicate with others, there are number of methods available for their communication and the most common method is sign language.
American Sign Language is considered as the de-facto standard of sign languages taught all over the world.
Sign Language consists of three major components:

a. **Finger Spelling:** Letter-by-Letter spelling of words during communication.
b. **Word-level Sign Vocabulary:** Majority of communication occurs via word-level signs.
c. **Expressions:** Facial, mouth or bodily expressions make up for some familiar words in communication.

OBJECTIVES:

This project aims at classifying the several sign language characters (Alphabets only) from a set of randomised sign language images using six different Machine Learning Models:

1. **Linear Regression**
2. **Logistic Regression**
3. **Support Vector Machine**
4. **Random Forest**
5. **Convolutional Neural Network**
6. **Artificial Neural Network**

The best accuracies from the combination of these models and respective test sizes were then compared.

DATASET INFORMATION:

The dataset consisting of the randomised images of American Sign Language characters were converted into the csv dataset files titled:

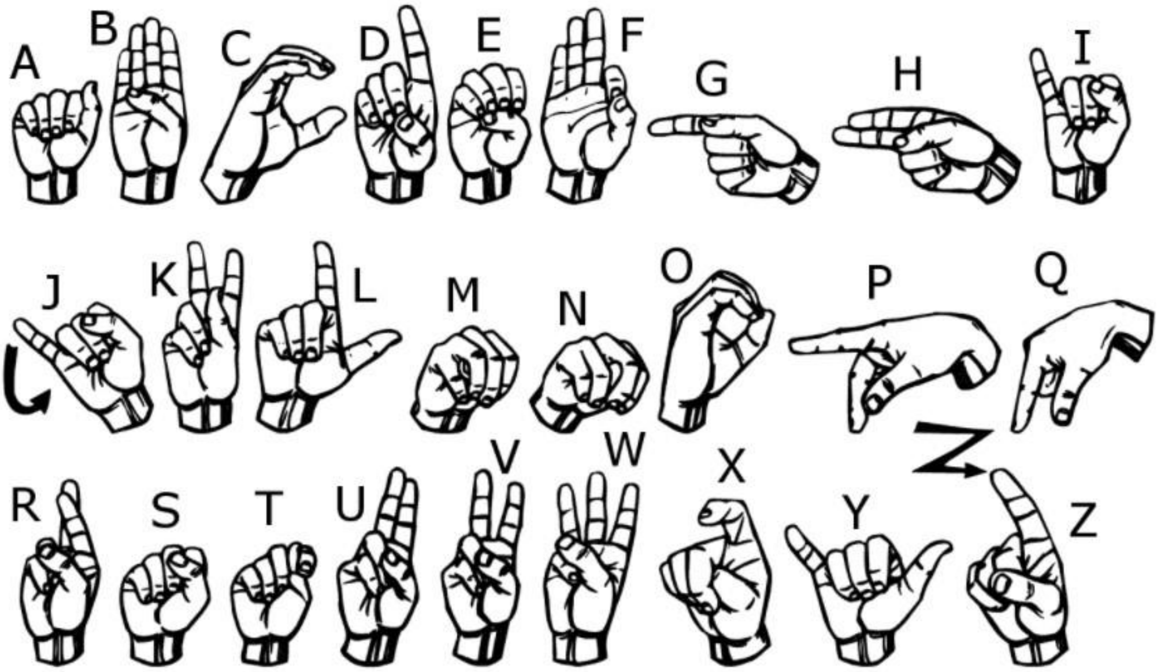
1. **ASL_train.csv (Training dataset)**
2. **ASL_test.csv (Testing dataset)**

 # Number of Classes = 25 (0-24)

[The dataset does not contain any symbols for 'J' and 'Z' as they are visually interpreted by motion of the hands similar to 'I' and 'D' respectively]

 # Number of Features = 784

The dataset consists of twenty-five distinct categories have been considered for English Alphabets (A-Z), ranging from (0 to 24)



(Representation of the alphabets in American Sign Language)

LINEAR REGRESSION MODEL:

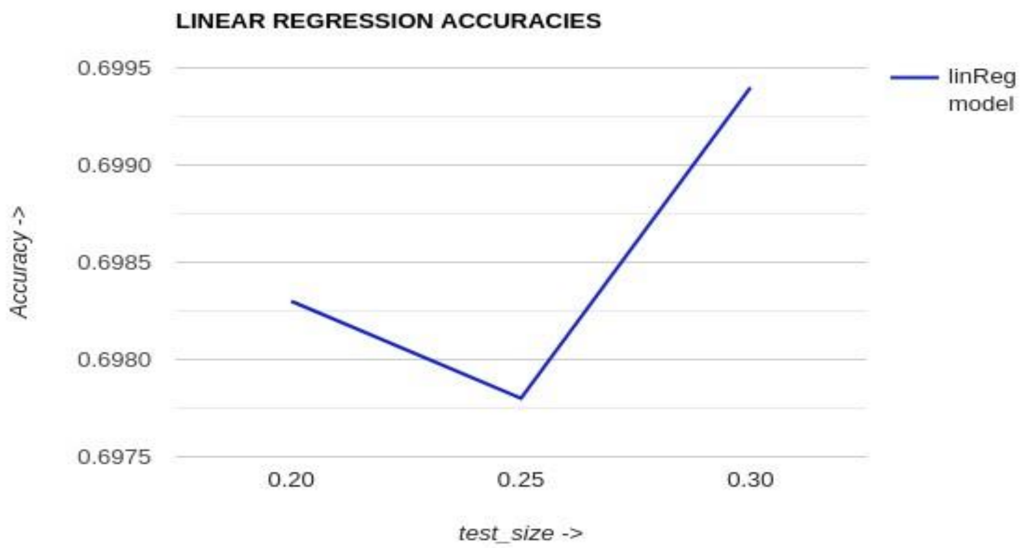
I created a **Linear Regression model** on the dataset with varying **test sizes** of **0.20**, **0.25** and **0.30**. Dataset consisting of many features (784), a low accuracy of around **~70%** was observed.

Error Metric: **Mean-Squared Error (MSE)**

Observations:

>> Test size of 0.20 provided an accuracy of 0.6983 with a Meansquared error of 16.04.
>> Test size of 0.25 provided an accuracy of 0.6978 with a Meansquared error of 16.12.
>> Test size of 0.30 provided an accuracy of 0.6994 with a Meansquared error of 16.02.
>> Maximum accuracy was found to be 0.6994 at a test size of 0.30.

Accuracies vs test size plot:



LOGISTIC REGRESSION MODEL:

I created a **Logistic Regression model** on the dataset with varying **test sizes** of **0.20**, **0.25** and **0.30**. Dataset consisting of many features (784), a mid-range accuracy of around **~87%** was observed.

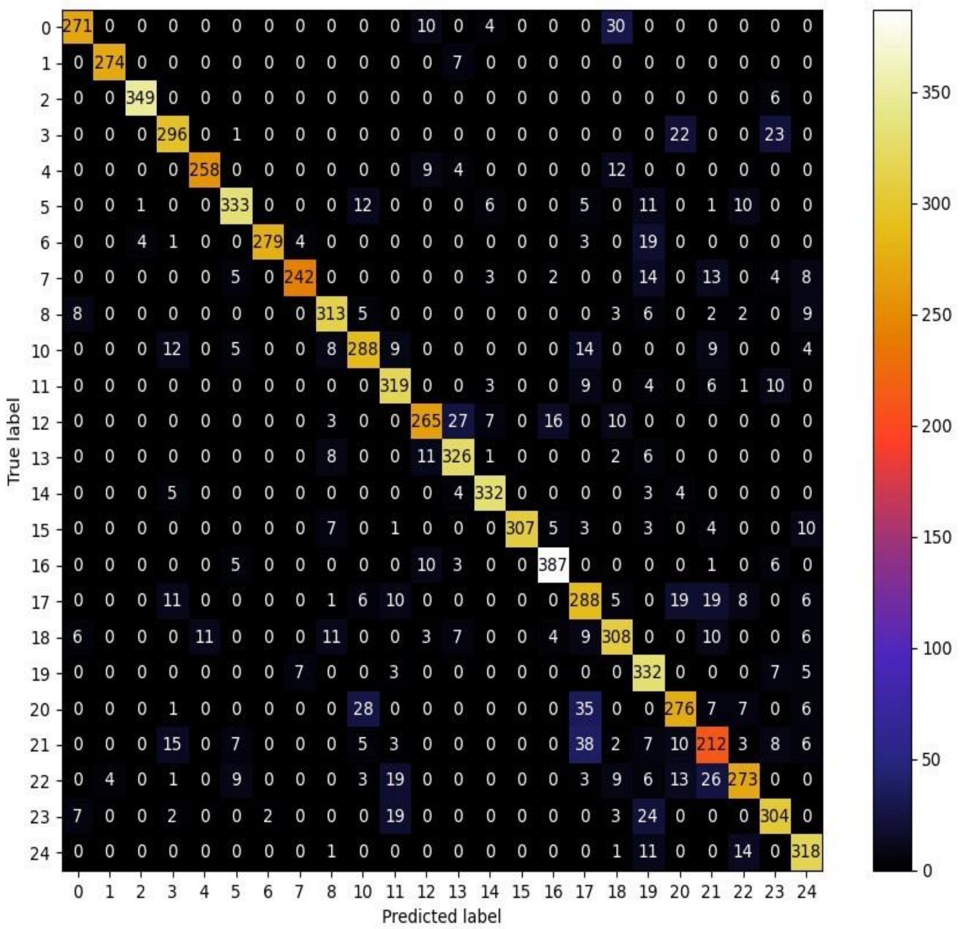
Solver Used: Saga (preferred for large datasets)
['**lbfgs**' and '**liblinear**' solvers produced insufficient accuracies]

Observations:

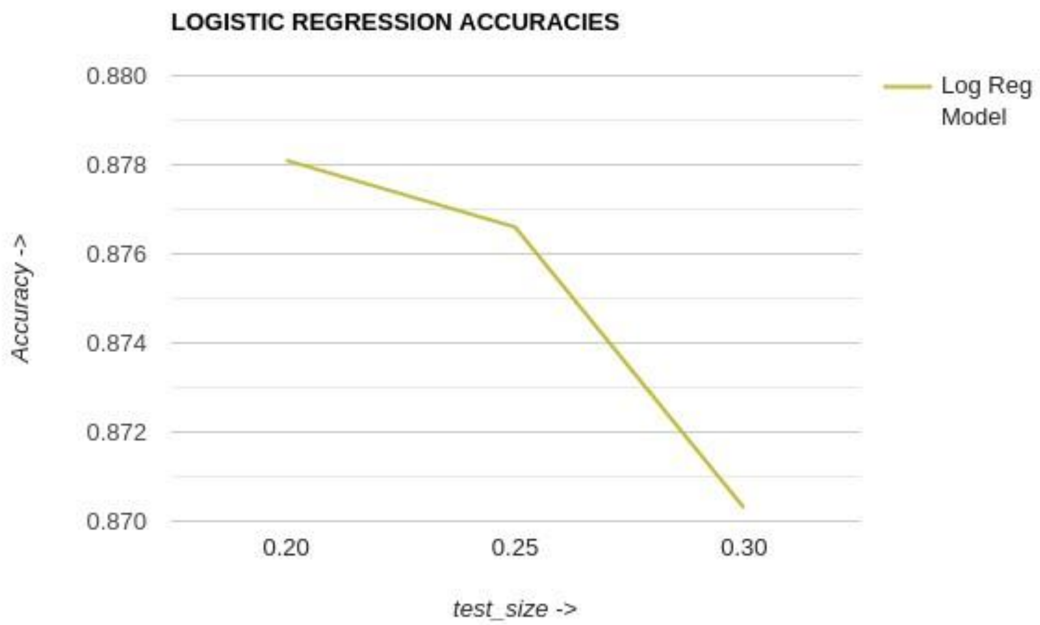
>> Test size of 0.20 provided an accuracy of 0.8781.
>> Test size of 0.25 provided an accuracy of 0.8766.
>> Test size of 0.30 provided an accuracy of 0.8703.
>> Maximum accuracy was found to be 0.8781 at a test size of 0.20.

Confusion Matrix:

LOGISTIC REGRESSION



Accuracies vs test size plot:



SUPPORT VECTOR MACHINE MODEL:

I created a **Support Vector Machine model** on the dataset with varying **test sizes** of **0.20**, **0.25** and **0.30**. Dataset consisting of a large number of features (784), an extremely high accuracy of around **~99%** was observed, which was discovered to be the case of **Over-fitting**.

Shape of Decision Function: **One-vs-one (ovo)**

Gamma Value: **0.0001**

One-vs-one ('ovo') was chosen as the shape of the decision function due to the nature of dataset being multi-class classification and 'rbf' (radial basis function) as the kernel. Kernels ('linear' and 'poly') had high execution times (>35 min).

Any gamma value > pow(10,-3) had extremely high runtimes. The decision boundary is observed to have lesser curvature with gamma < pow(10,-3).

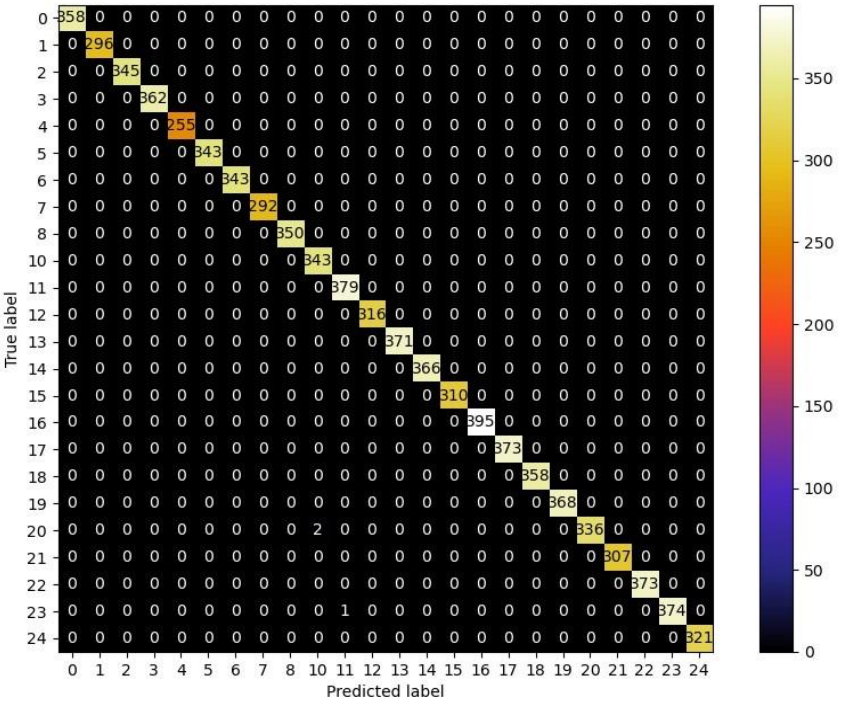
Observations:

>> Test size of 0.20 provided an accuracy of 0.9998.
>> Test size of 0.25 provided an accuracy of 0.9972.
>> Test size of 0.30 provided an accuracy of 0.9969.
>> Maximum accuracy was found to be 0.9998 at a test size of 0.20.

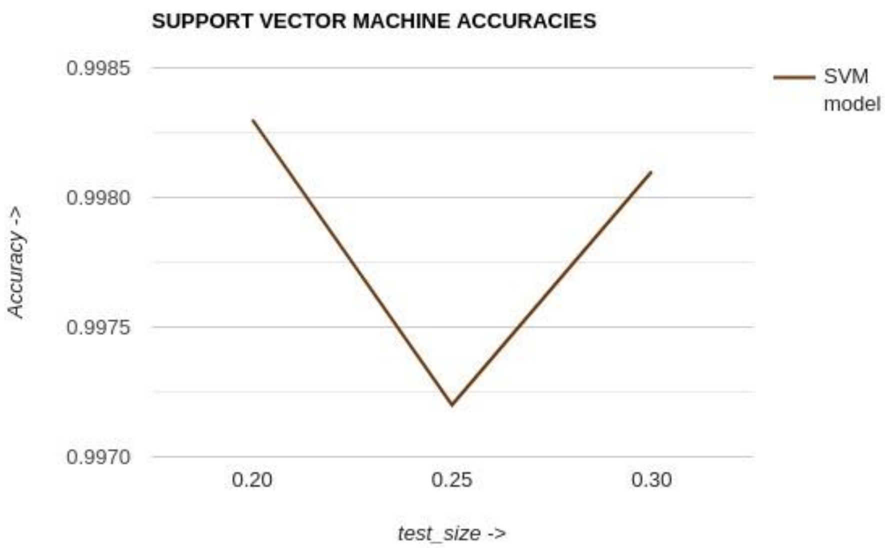
One-vs-one ('ovo') was chosen as the shape of the decision function due to the nature of dataset being multiclass classification and 'rbf' (radial basis function) as the kernel. Kernels ('linear' and 'poly') had high execution times (>35 min on local machine).

Any gamma value > pow(10,-3) had extremely high runtimes. The decision boundary is observed to have lesser curvature with gamma < pow(10,-3).

Confusion Matrix: **SUPPORT VECTOR MACHINE**



Accuracies vs test_size plot:



RANDOM FOREST MODEL:

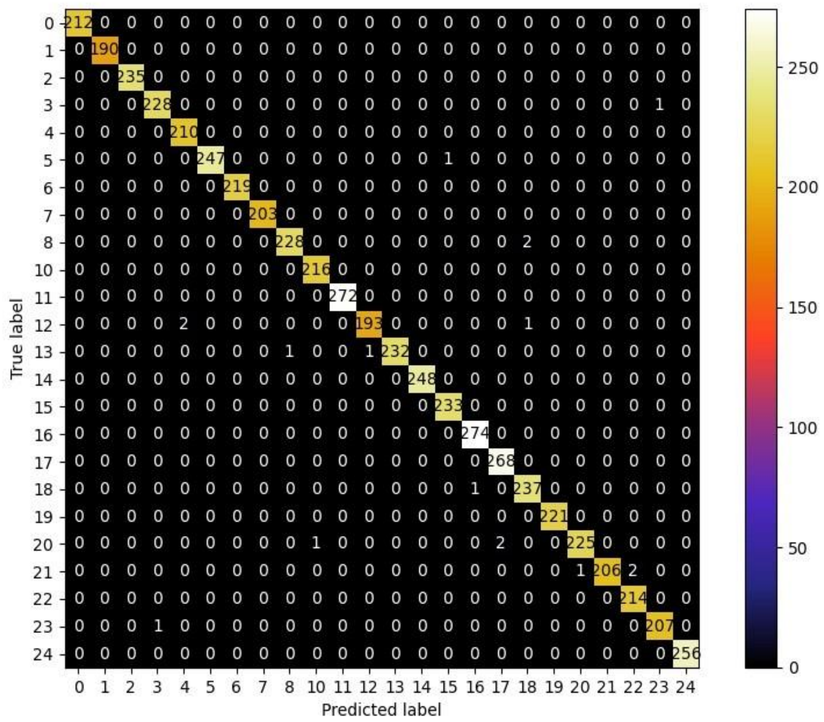
I created a **Random Forest model** on with **100 estimators** and varying **test sizes** of **0.20**, **0.25** and **0.30**. Dataset consisting of a large number of features (784), an extremely high accuracy of around **~99%** was observed. This model with the help of 100 Decision Trees used averaging to improve the predictive accuracy and solved the problem of **Overfitting** observed in **SVM model**.

Observations:

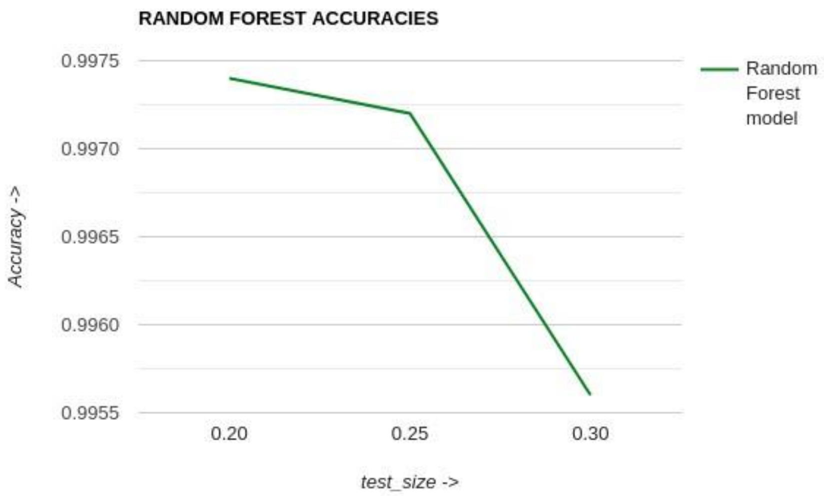
>> Test size of 0.20 provided an accuracy of 0.9974.
>> Test size of 0.25 provided an accuracy of 0.9972.
>> Test size of 0.30 provided an accuracy of 0.9956.

>> Maximum accuracy was found to be 0.9998 at a test size of 0.20.

Confusion Matrix: **RANDOM FOREST MODEL**



Accuracies vs test_size plot:



The random forests model calculates a response variable by creating many different decision trees and then putting each object to be modeled down each of the decision trees, which is determined by evaluating the responses from all of the trees.

This model with the help of 100 Decision Trees used averaging to improve the predictive accuracy and solved the problem of overfitting observed in SVM model with an accuracy of ~99.6%

Convolutional Neural Network (CNN)

I created a **Convolutional Neural Network model** on the dataset that comprised of **three Convolution Layers** (consisting of **128**, **64** and **32 units** respectively) and **three MaxPooling Layers** (of **window-size 3*3** , **2*2** and **2*2** respectively).

Activation Function: ReLU

Model Summary:

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 128)	3328
max_pooling2d (MaxPooling2D)	(None, 14, 14, 128)	0
conv2d_1 (Conv2D)	(None, 14, 14, 64)	32832
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 64)	0
conv2d_2 (Conv2D)	(None, 7, 7, 32)	8224
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 32)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 512)	262656
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 24)	12312

Total params: 319,352
Trainable params: 319,352
Non-trainable params: 0

Observations:

>> Accuracy was found to be 0.9297267198 after 35 epochs of batch_size 138.

The benefit derived by using CNNs here is due to their ability to develop an internal representation of a two-dimensional image, the model learns the position and scale in variant structures in the data, which is extremely essential while dealing with images.

Artificial Neural Network (ANN)

I created an **Artificial Neural Network model** on the dataset with varying **test sizes** of **0.20**, **0.25** and **0.30**. The ANN model consisted of:

1. A ***"Dense"*** layer of 300 units followed by a ***"Dropout"*** layer of **rate 0.3**. **(*Activation function: ReLU*)**
2. Three sets of ***"Dense"*** layers consisting of 100 units each.**(*Activation function: ReLU*)**
3. A final ***"Dense"*** layer with 25 units. **(*Activation function: Softmax*)**

Dataset consisting of a large number of features (784),a high accuracy of around **~97%** was observed.

Activation Functions: ReLU, Softmax

<u>Loss Metric:</u> Categorical Cross-Entropy *(multi-class classification problem)* **<u>Optimizer:</u> Nadam** *(provided the optimal accuracy)*

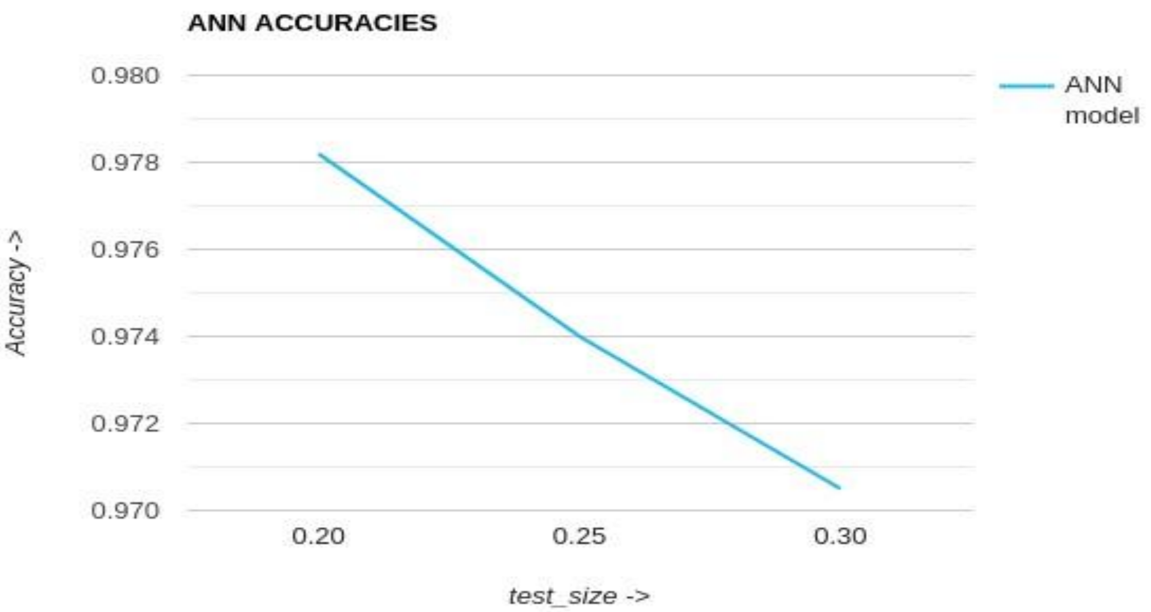
Model Summary:

```
model = Sequential(Flatten(input_shape = [28, 28]))
model.add(Dense(300, activation='relu'))
model.add(Dropout(rate= 0.3))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(100, activation='relu'))
model.add(Dense(25,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy', optimizer='nadam',    metrics='accuracy')
h = model.fit(X_train, y_train, epochs=6, verbose=True)
```

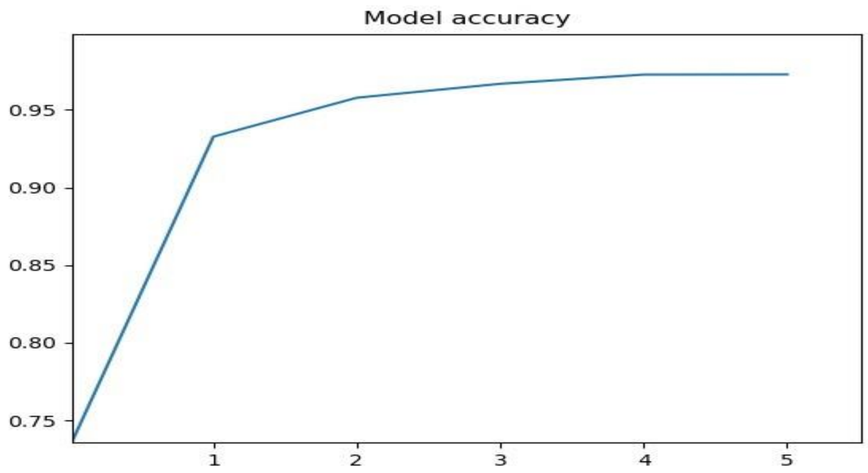
Observations:

>> Test size of 0.20 provided an accuracy of 0.9781.
>> Test size of 0.25 provided an accuracy of 0.9740.
>> Test size of 0.30 provided an accuracy of 0.9705.
>> Maximum accuracy was found to be 0.9781 at a test size of 0.20 after 6 epochs.

Accuracies vs test_size plot:



Accuracies vs Epoch plot:

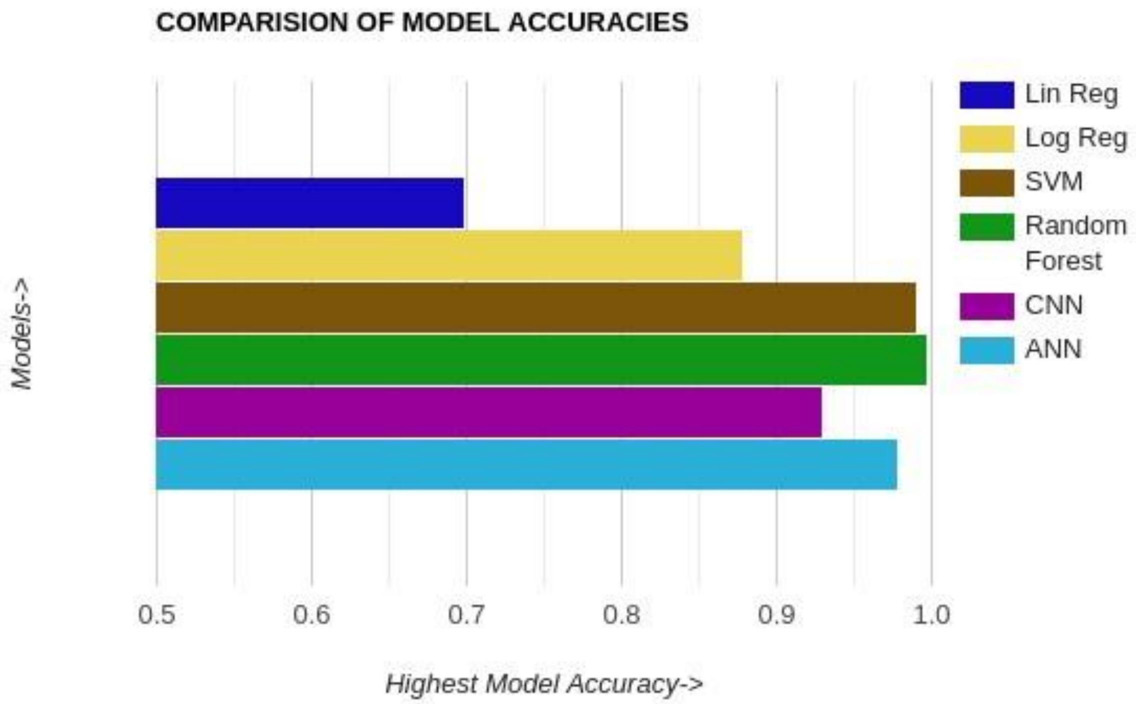


MODEL COMPARISONS: **(Models w.r.t. test sizes)**

Model Accuracy:	test_size = 0.20	test_size = 0.25	test_size = 0.30
Linear Regression	0.6983	0.6978	0.6994
Logistic Regression	0.8781	0.8766	0.8703
SVM	0.9998	0.9972	0.9969
Random Forest	0.9974	0.9972	0.9956
ANN	0.9781	0.9740	0.9705

>> Maximum accuracy was found out from the SVM Model with test size of 0.20 (OVERFITTING).
>> Minimum accuracy was found out from the Linear Regression Model with test size of 0.20.

Comparative Representation of the Model Accuracies:



REFERENCES:

1.American Sign Language Character Recognition Using Convolution Neural Network (https://link.springer.com/chapter/10.1007/978-981-10-5547-8_42)
2.https://data-flair.training/blogs/sign-language-recognition-python-ml-opencv/
3.https://debuggercafe.com/american-sign-language-detection-using-deep-learning/
4.https://stats.stackexchange.com/questions/52773/what-can-cause-pca-to-worsen-results-of-a-classifier.

****