Compare Support Vector Machines to a 3 layer Neural Networks on the titanic dataset

N N Kundan
Machine Learning Engineer Intern
AI Technology & Systems
California, United States
www.ai-techsystems.com

nnkundan2018@gmail.com

***Abstract*— This report is about the detail comparison of 3 Layer neural network model and Support Vector machine algorithm on the titanic dataset. The titanic dataset is used from Kaggle. Jupyter Notebook is used for writing codes which is available in Anaconda platform.**

***Index Terms*—Machine Learning, Deep Learning, Support Vector Machine (SVM), Neural Network.**

I. INTRODUCTION

In this project, we can see a comparison of 3 Layer neural network model and Support Vector machine algorithm on the titanic dataset. Here, we have to find out which model gives better accuracy. In machine learning, support-vector machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis whereas neural networks are computing systems that are inspired by, but not identical to, biological neural networks. Such systems "learn" to perform tasks by considering examples, generally without programmed with task-specific rules.

II. DATASET INFORMATION

In this project, we have used titanic dataset from Kaggle. The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.
One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.
In this Project, we have to predict which passengers survived the tragedy.
The data has been split into two groups:

- Training set(train.csv)
- Test set (test.csv)

The training set should be used to build our machine learning models. For the training set, outcome (also known as the "ground truth") for each passenger is provided.
The test set should be used to see how well our model performs on unseen data. For the test set, the ground truth for each passenger is not provided. We have to predict these outcomes. For each passenger in the test set, we use the trained model to predict whether or not passengers survived the sinking of the Titanic.
Note:

For Survival, 0 = No, 1 = Yes.

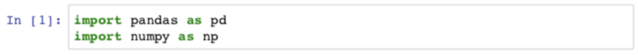
A. LOADING DATASET INTO PANDAS DATAFRAME

Pandas is an open-source, Berkeley Software Distribution (BSD) – licensed library providing high-performance, easy-to-ease data structures and data analysis tools for the Python programming language.

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high level mathematical functions to operate on these arrays.

To use *pandas* and *numpy* library, we need to import pandas and numpy module into the environment.

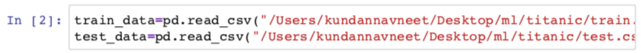
```
import pandas as pd

import numpy as pd
```



```
In [1]: import pandas as pd
        import numpy as np
```

Pandas library has provided different methods for loading datasets with many different formats onto DataFrames. Here we use:

read_csv to read comma separated values.

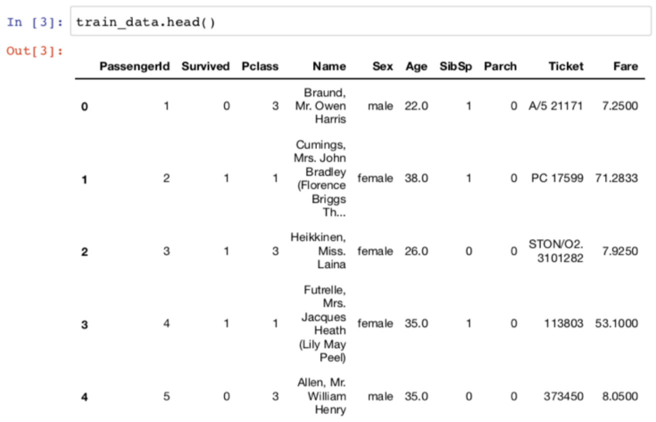


```
In [2]: train_data=pd.read_csv("/Users/kundannavneet/Desktop/ml/titanic/train.
        test_data=pd.read_csv("/Users/kundannavneet/Desktop/ml/titanic/test.cs
```

train.csv is loaded in DataFrame variable named train_data and test.csv is loaded in DataFrame variable named test_data.

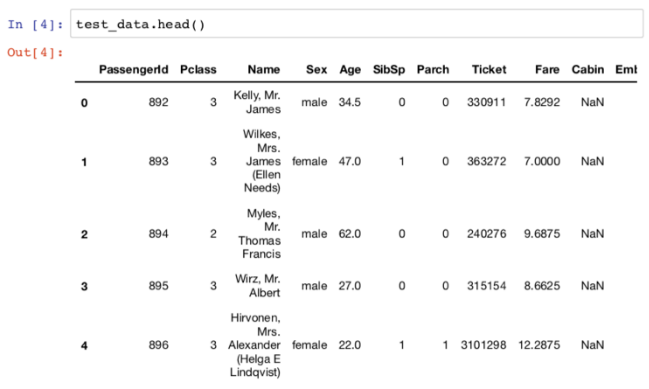
B. DISPLAYING FIRST FEW RECORDS OF THE DATAFRAME

To display the first few rows from the DataFrame, use function *head( )*



```
In [3]: train_data.head()
Out[3]:
```

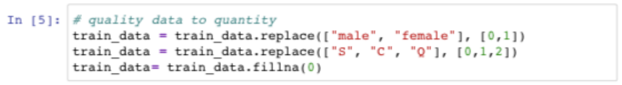
	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500



```
In [4]: test_data.head()

Out[4]:
```

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Emt
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	

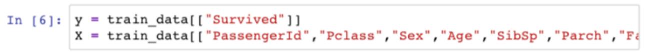
C. CONVERT CATEGORICAL VALUE INTO NUMERIC VALUE AND HANDLING OF MISSING DATA



```
In [5]: # quality data to quantity
        train_data = train_data.replace(["male", "female"], [0,1])
        train_data = train_data.replace(["S", "C", "Q"], [0,1,2])
        train_data= train_data.fillna(0)
```

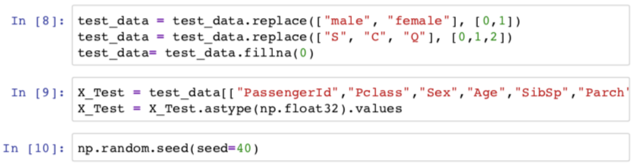
In Sex column we assign ‘male’ as 0 and ‘female’ as 1. Similarly in Embarked column we assign ‘S’ as 0, ‘C’ as 1, ‘Q’ as 2.
For missing values: Replace all NaN values with 0's in Pandas DataFrame.

D. CREATE FEATURE SET(X) AND OUTCOME(Y)



```
In [6]: y = train_data[["Survived"]]
        X = train_data[["PassengerId","Pclass","Sex","Age","SibSp","Parch","Fa
```

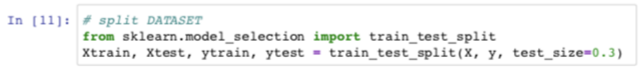
E. DATA PREPROCESSING FOR TEST FILE



```
In [8]: test_data = test_data.replace(["male", "female"], [0,1])
        test_data = test_data.replace(["S", "C", "Q"], [0,1,2])
        test_data= test_data.fillna(0)

In [9]: X_Test = test_data[["PassengerId","Pclass","Sex","Age","SibSp","Parch"
        X_Test = X_Test.astype(np.float32).values

In [10]: np.random.seed(seed=40)
```

Here we have prepared test data according to our desired need for prediction.

F. SPLITTING THE DATASET INTO TRAINING AND VALIDATION SETS.



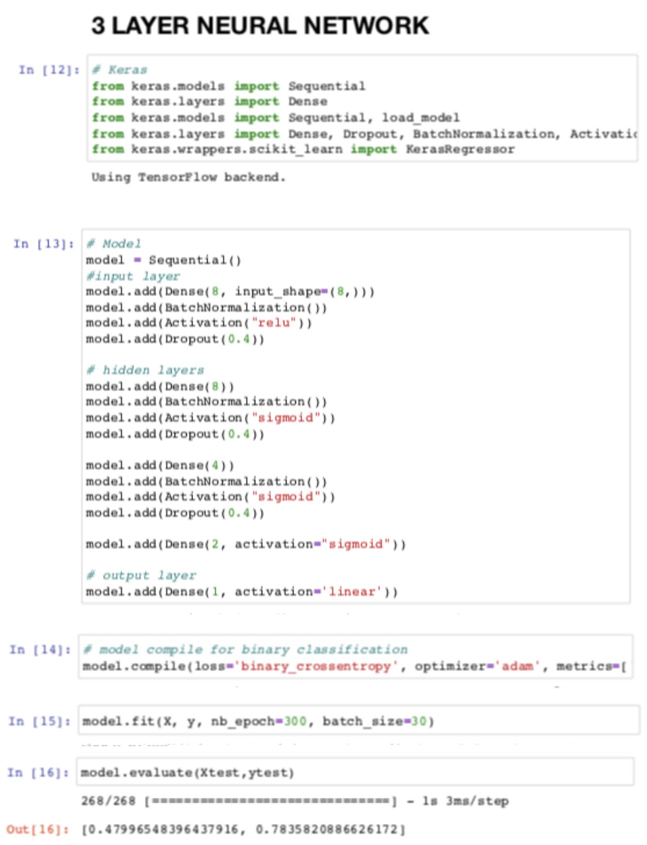
```
In [11]: # split DATASET
         from sklearn.model_selection import train_test_split
         Xtrain, Xtest, ytrain, ytest = train_test_split(X, y, test_size=0.3)
```

train_test_split( ) function from sklearn.model_selection module provides the ability to split the dataset randomly into training and validation datasets.
train_test_split( ) returns four variables.

- *xtrain* contain *x* feature of the training set.
- *ytrain* contains the values of response variable for the training set.
- *xtest* contains *x* features of the test set.
- *ytest* contains the values of response variable for the test set.

test_size = 0.3 implies 30% of the data is used for validating the model and remaining 70% is used for training the model.

G. NEURAL NETWORK MODEL

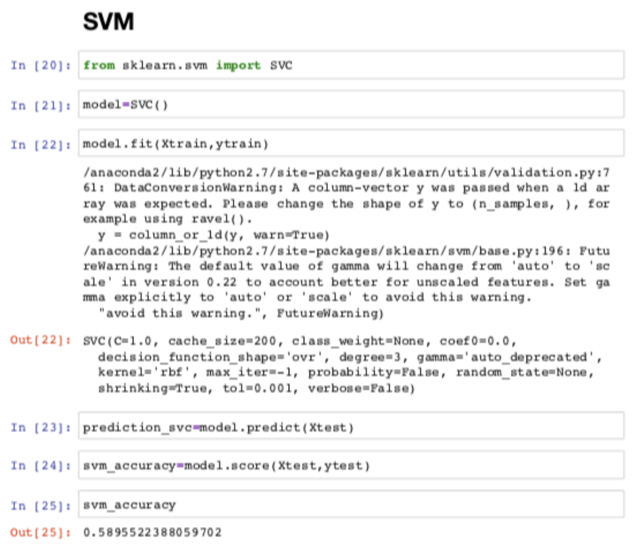


3 LAYER NEURAL NETWORK

```
In [12]: # Keras
         from keras.models import Sequential
         from keras.layers import Dense
         from keras.models import Sequential, load_model
         from keras.layers import Dense, Dropout, BatchNormalization, Activatic
         from keras.wrappers.scikit_learn import KerasRegressor

         Using TensorFlow backend.

In [13]: # Model
         model = Sequential()
         #input layer
         model.add(Dense(8, input_shape=(8,)))
         model.add(BatchNormalization())
         model.add(Activation("relu"))
         model.add(Dropout(0.4))

         # hidden layers
         model.add(Dense(8))
         model.add(BatchNormalization())
         model.add(Activation("sigmoid"))
         model.add(Dropout(0.4))

         model.add(Dense(4))
         model.add(BatchNormalization())
         model.add(Activation("sigmoid"))
         model.add(Dropout(0.4))

         model.add(Dense(2, activation="sigmoid"))

         # output layer
         model.add(Dense(1, activation='linear'))

In [14]: # model compile for binary classification
         model.compile(loss='binary_crossentropy', optimizer='adam', metrics=[

In [15]: model.fit(X, y, nb_epoch=300, batch_size=30)

In [16]: model.evaluate(Xtest,ytest)

         268/268 [==============================] - 1s 3ms/step

Out[16]: [0.47996548396437916, 0.7835820886626172]
```

First we have to import essential library for neural network model, then we define our model and created input layer, hidden layer and output layer.
Then, we have compile our model for binary classification and the fit the model.
Finally, the 3 layer neural network model gives the accuracy of 78.35%.

H. SUPPORT VECTOR MACHINE

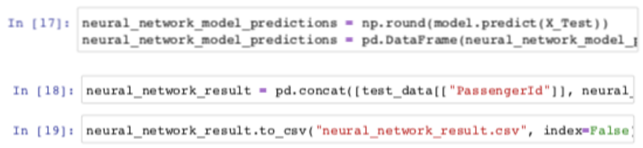


SVM

```
In [20]: from sklearn.svm import SVC

In [21]: model=SVC()

In [22]: model.fit(Xtrain,ytrain)

         /anaconda2/lib/python2.7/site-packages/sklearn/utils/validation.py:7
         61: DataConversionWarning: A column-vector y was passed when a 1d ar
         ray was expected. Please change the shape of y to (n_samples, ), for
         example using ravel().
           y = column_or_1d(y, warn=True)
         /anaconda2/lib/python2.7/site-packages/sklearn/svm/base.py:196: Futu
         reWarning: The default value of gamma will change from 'auto' to 'sc
         ale' in version 0.22 to account better for unscaled features. Set ga
         mma explicitly to 'auto' or 'scale' to avoid this warning.
           "avoid this warning.", FutureWarning)

Out[22]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
           decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
           kernel='rbf', max_iter=-1, probability=False, random_state=None,
           shrinking=True, tol=0.001, verbose=False)

In [23]: prediction_svc=model.predict(Xtest)

In [24]: svm_accuracy=model.score(Xtest,ytest)

In [25]: svm_accuracy

Out[25]: 0.5895522388059702
```

Here we have to import SVC from sklearn.svm library. After that we have to create our model and then we have to fit our model. Support vector machine model gives the accuracy of 58.95%.

I. FILE SUBMISSION



```
In [17]: neural_network_model_predictions = np.round(model.predict(X_Test))
         neural_network_model_predictions = pd.DataFrame(neural_network_model_

In [18]: neural_network_result = pd.concat([test_data[["PassengerId"]], neural_

In [19]: neural_network_result.to_csv("neural_network_result.csv", index=False
```

test.csv is the original file in which we have to predict which passenger have survived and which have not survived. Initially we have stored this file in test_data named Dataframe.

3 layer neural network gives better accuracy than support vector machine so we have predicted passenger survival using 3 layer neural network.

File in which we have predicted passenger survival is of name neural_network_result.csv

J. CONCLUSION

- *3 layer neural network accuracy=78.35%*
- *support vector machine accuracy= 58.95%.*

In my project 3 layer neural network gives better accuracy than support vector machine.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to team AITS for providing me this opportunity to make this project. Also, I would like to thank to Kaggle.com for providing titanic dataset.

REFERENCES

[1] pandas Library: https://pandas.pydata.org

[2] NumPy Library: https://numpy.org

[3] Scikit-learn documentation at : https://scikit-learn.org

[4] keras : https://keras.io

[5] ‘Machine Learning using python’ by Manaranjan Pradhan and U Dinesh kumar, Wiley India Pvt ltd.