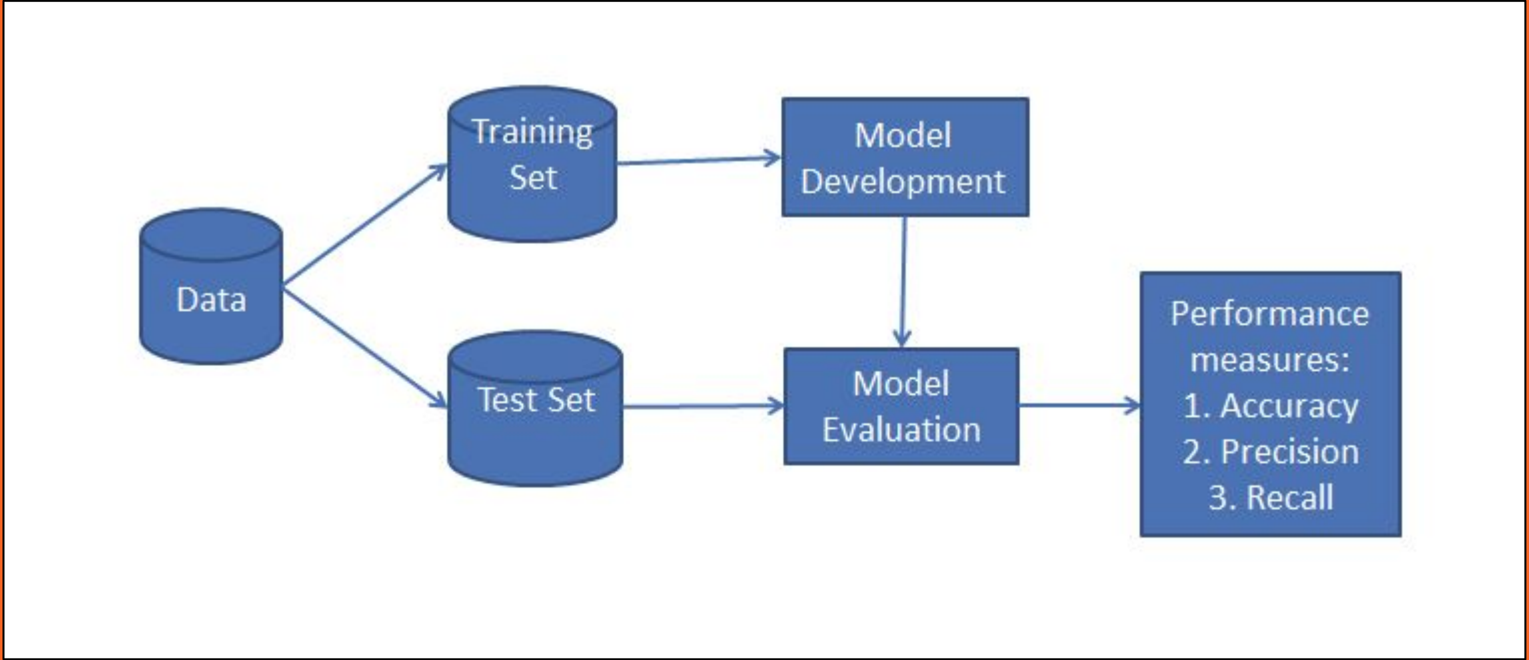
YASH RANDIVE

Restaurant Rating Prediction Using Sentiment Analysis and Machine Learning

Part III-b

YASH RANDIVE

QUICK RECAP



YASH
RANDIVE

Predictive Modelling

- ➔ Support Vector Machine Classifier
- ➔ Deep Neural Network(DNN)

YASH
RANDIVE

Modelling Workflow

1. Import relevant Libraries for the Model
2. Create the Model and fit the training data(x_train) to it
3. Evaluate the model by making predictions on the trained model using the x_test model testing data
4. Examine the Confusion Matrix and calculate the accuracy
5. Save the predictions in a .csv file

The Support Vector Machine Classifier

1. Import relevant Libraries for the Model
2. Create the SVM Model and fit the training data(x_train) to it
3. Evaluate the model by making predictions on the trained model using the x_test model testing data
4. Examine the Confusion Matrix and calculate the accuracy
5. Save the predictions in a .csv file

YASH
RANDIVE

SVMs

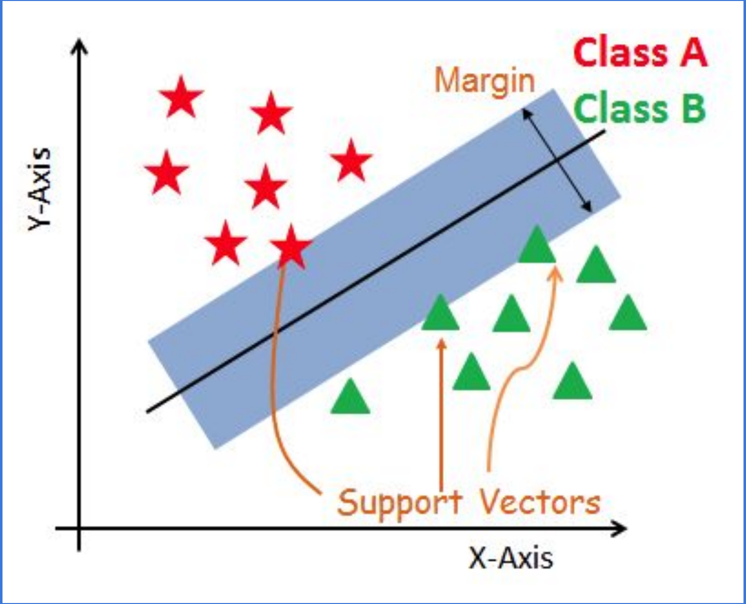
1. Can be employed in classification and regression problems
2. It can easily handle multiple continuous categorical variables
3. The core idea of an SVM is to find a maximum marginal hyperplane(MMH) that best divides the dataset into classes

Hyperplane

A hyperplane is a decision plane which separates between a set of objects having different class memberships

Margin

A margin is a gap between the two lines of the closest class points. This is calculated as the perpendicular distance from the line to support vectors(or closest points). Notably, if the margin is larger between the classes it is considered as a good margin

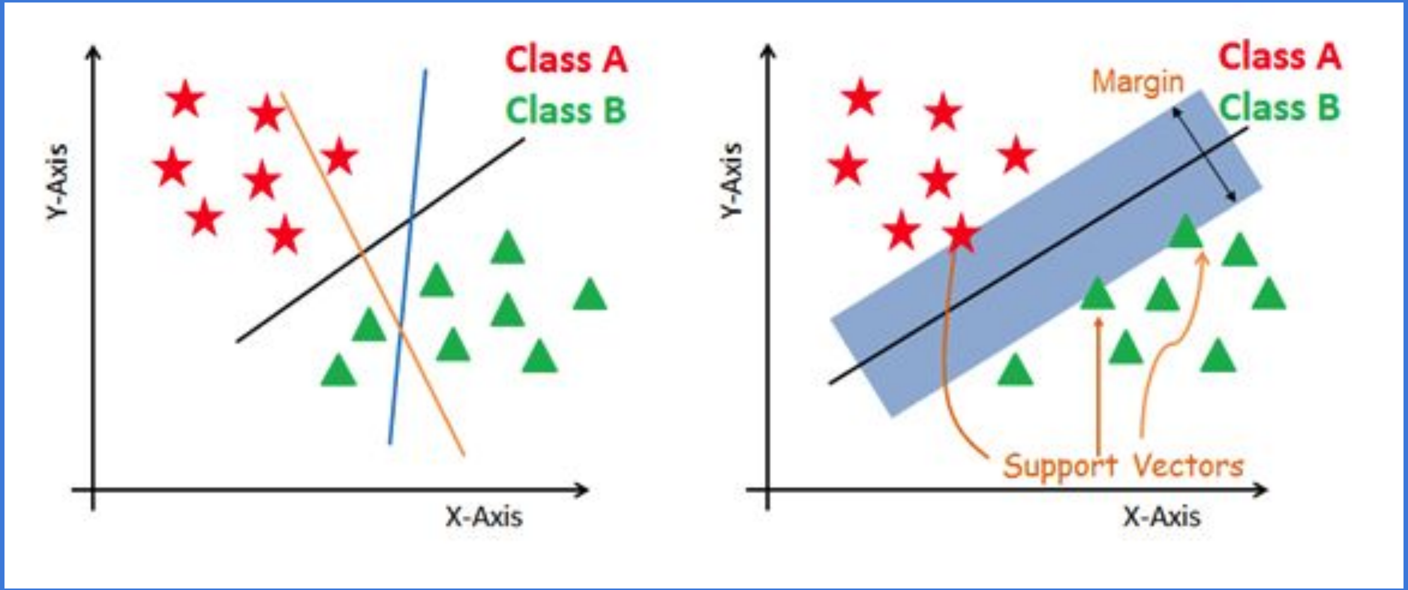


How does SVM work?

The main objective is to segregate the given dataset in the best possible way. The distance between the either nearest points is known as the margin. The objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset

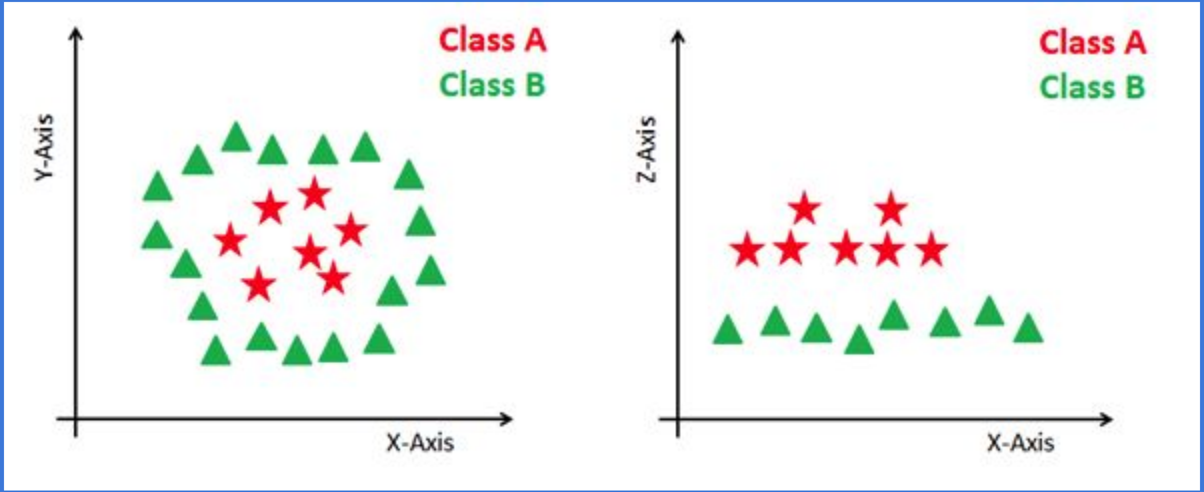
How does SVM work?

YASH RANDIVE



1. Generate Hyperplanes which segregate the classes in the best way
2. Select the right hyperplane with the maximum segregation from the nearest data points

What about curved data?



1. Such data cannot be classified using a “Linear Hyperplane”
2. SVM uses a ”Kernel Trick” to transform the input space into a higher dimensional space i.e. the Data is plotted on the X and Z axis (as Z is the squared sum of both X and Y (Y = X^2 and Z = Y^2))

SVM Kernels

YASH RANDIVE

A kernel transforms an input data space into the required form. SVM uses a technique called Kernel Trick; here the kernel takes a low-dimensional input space and transforms it into a high dimensional input space

Types of Kernels:

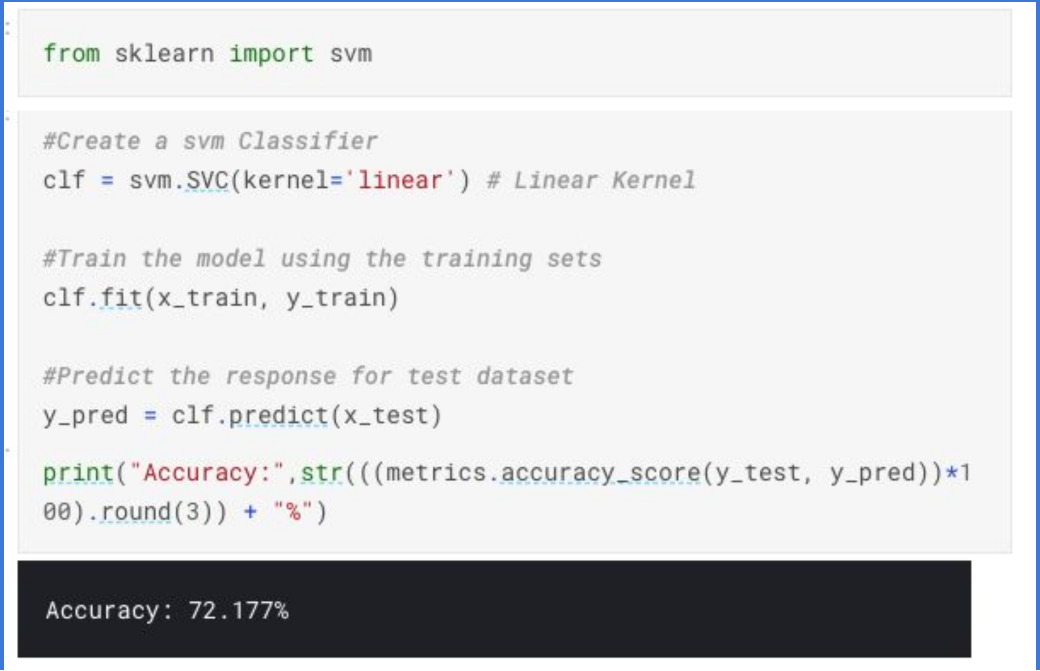
1. Linear Kernel : A linear kernel can be used as normal dot product any two given observations. The product between two vectors is the sum of the multiplication of each pair of input values.

   K(x, xi) = sum(x * xi)

2. Polynomial Kernel : The polynomial kernel can distinguish between curved or non-linear input space
3. Radial Basis Function : It is a popular kernel function, commonly used in support vector machine classification. RBF can map an input space in infinite dimensional space

SVM Implementation

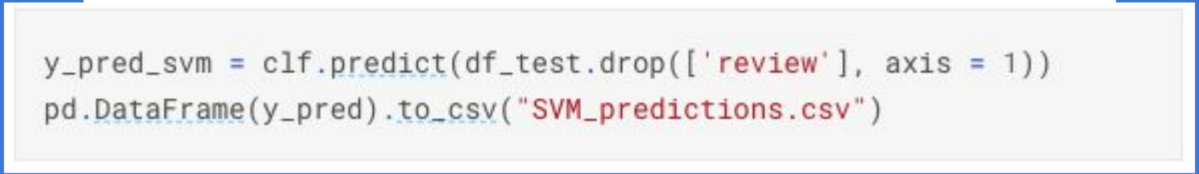
YASH
RANDIVE



```
from sklearn import svm
```

```
#Create a svm Classifier
clf = svm.SVC(kernel='linear') # Linear Kernel

#Train the model using the training sets
clf.fit(x_train, y_train)

#Predict the response for test dataset
y_pred = clf.predict(x_test)

print("Accuracy:",str(((metrics.accuracy_score(y_test, y_pred))*1
00).round(3)) + "%")
```

```
Accuracy: 72.177%
```

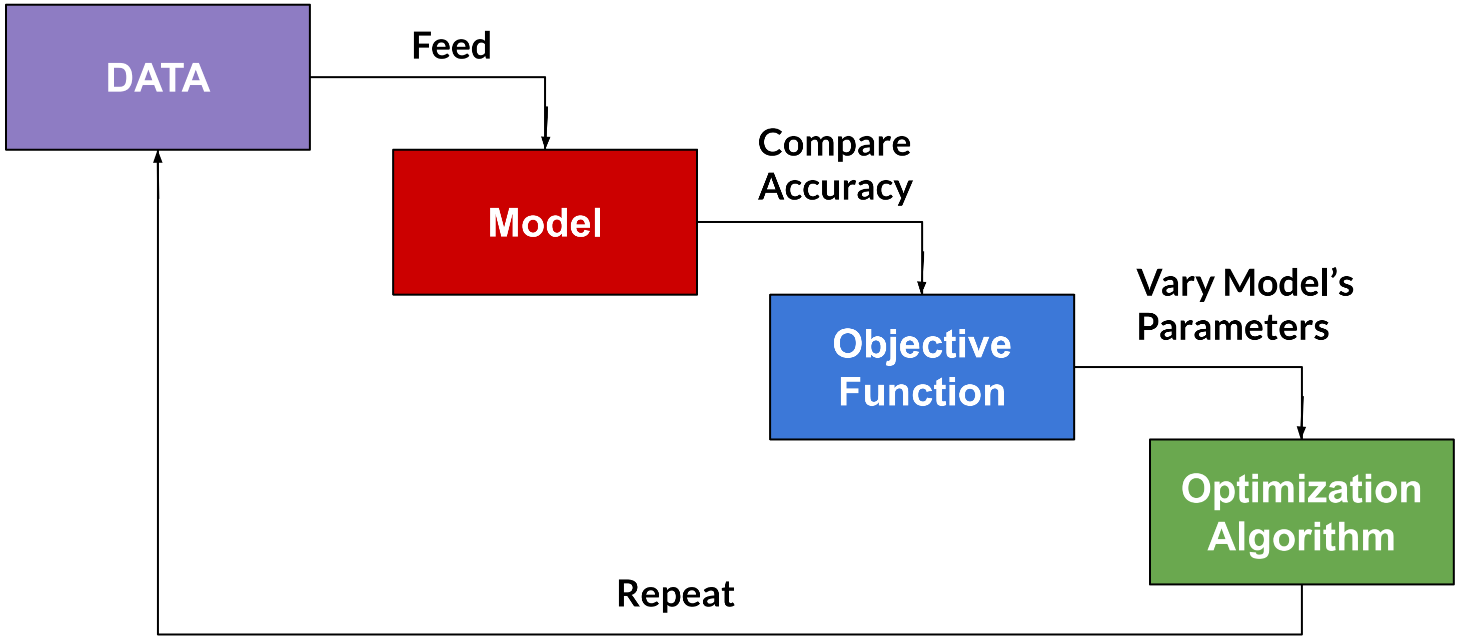


```
y_pred_svm = clf.predict(df_test.drop(['review'], axis = 1))
pd.DataFrame(y_pred).to_csv("SVM_predictions.csv")
```

YASH
RANDIVE

The Deep Neural Network Model

Fundamentals of the DNN Model



Fundamentals of the DNN Model

1. **Model :** The model contains all the underlying mathematical functions and logic on the Neural Network
2. **Objective Function:** The objective function is the measure used to evaluate how well the models outputs match the desired values
   a. **Loss Functions :** The lower the loss function the higher the accuracy of the model
   b. **Reward Functions :** Higher the reward function higher the accuracy of the model

Fundamentals of the DNN Model

Loss Functions : **The lower the loss function the higher the accuracy of the model**

Types of Loss Functions :

1. **L2- Norm Loss (For Regression)**
2. **Cross - Entropy (For Classification)**

# Note : Any function that holds the basic property, *“Higher for worse results, lower for better results is a loss function”*

Fundamentals of the DNN Model

3. Optimization Algorithm (Gradient Descent) : It is used to reduce the loss function of the model i.e. optimize the model in order to get the best accuracy

   a. Learning Rate : The rate at which the machine learning algorithm forgets old beliefs for new ones

Usually we want the Learning Rate to be High enough so that we can reach the closest minimum in a rational amount of time and Low Enough so that we don't oscillate around the minimum

TensorFlow Google

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

TensorFlow Google

TensorFlow DOES NOT work with .CSV or .XLXS files. It is Tensor Based, hence, it works with Tensors

- Tensors are nothing but multidimensional martices
- Tensors can be represented as n-dimensional arrays

TensorFlow Google

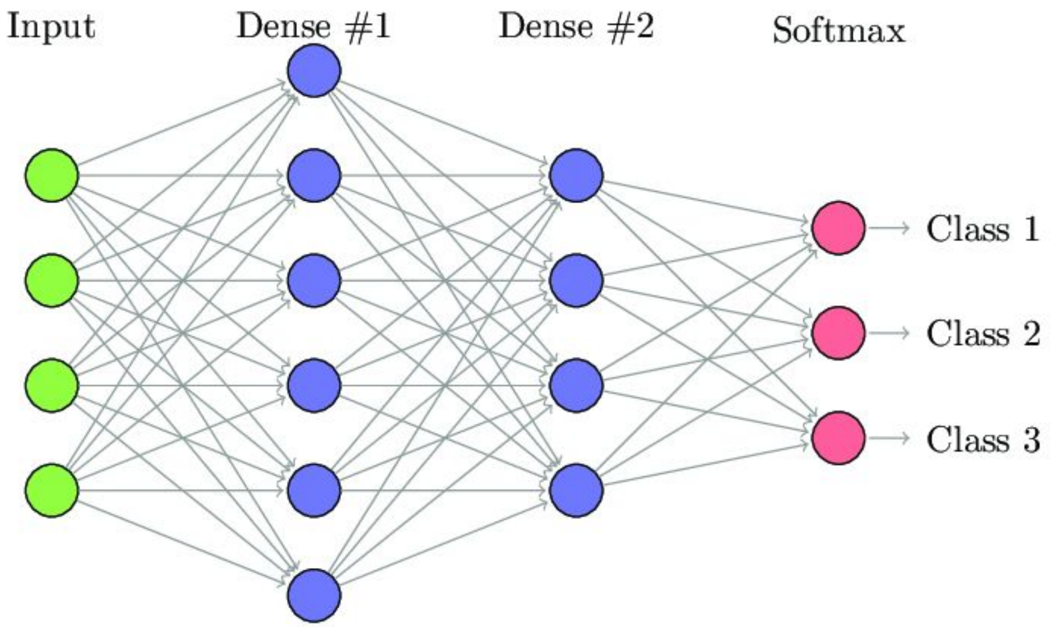
.npz files are NumPy's file type. It allows us to save ndarrays(n-dimensional arrays)

We can work with Tensorflow using these ndarrays.

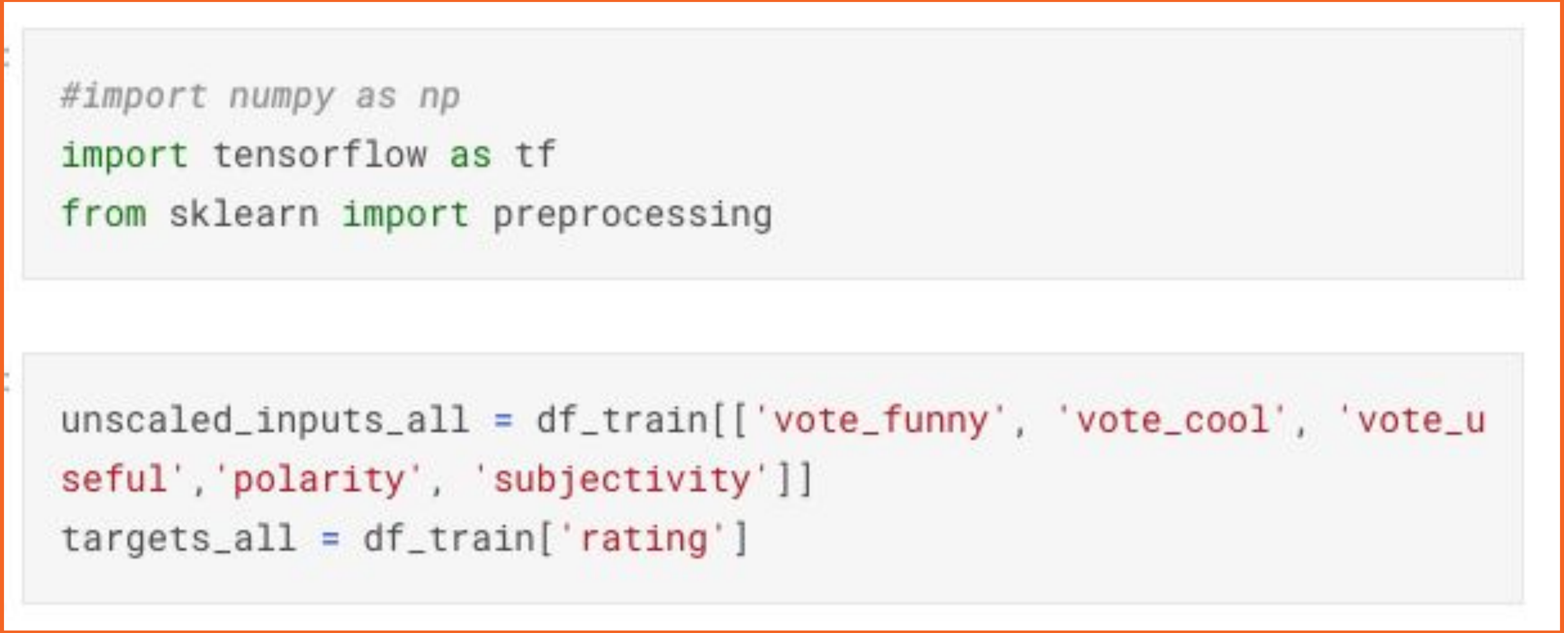
Model Workflow

1. Create new Unscaled Inputs and Targets Variables
2. Scale the inputs using *preprocessing.scale()*
3. Shuffling the Data
4. Splitting data into training and validation sets
5. Saving the Data as .npz files
6. Loading the .npz Files
7. Outlining the Model
8. Fitting the Model
9. Saving the Model Weights
10. Visualising the Training
11. Making predictions on the df_test test data and saving the predictions

Model Workflow



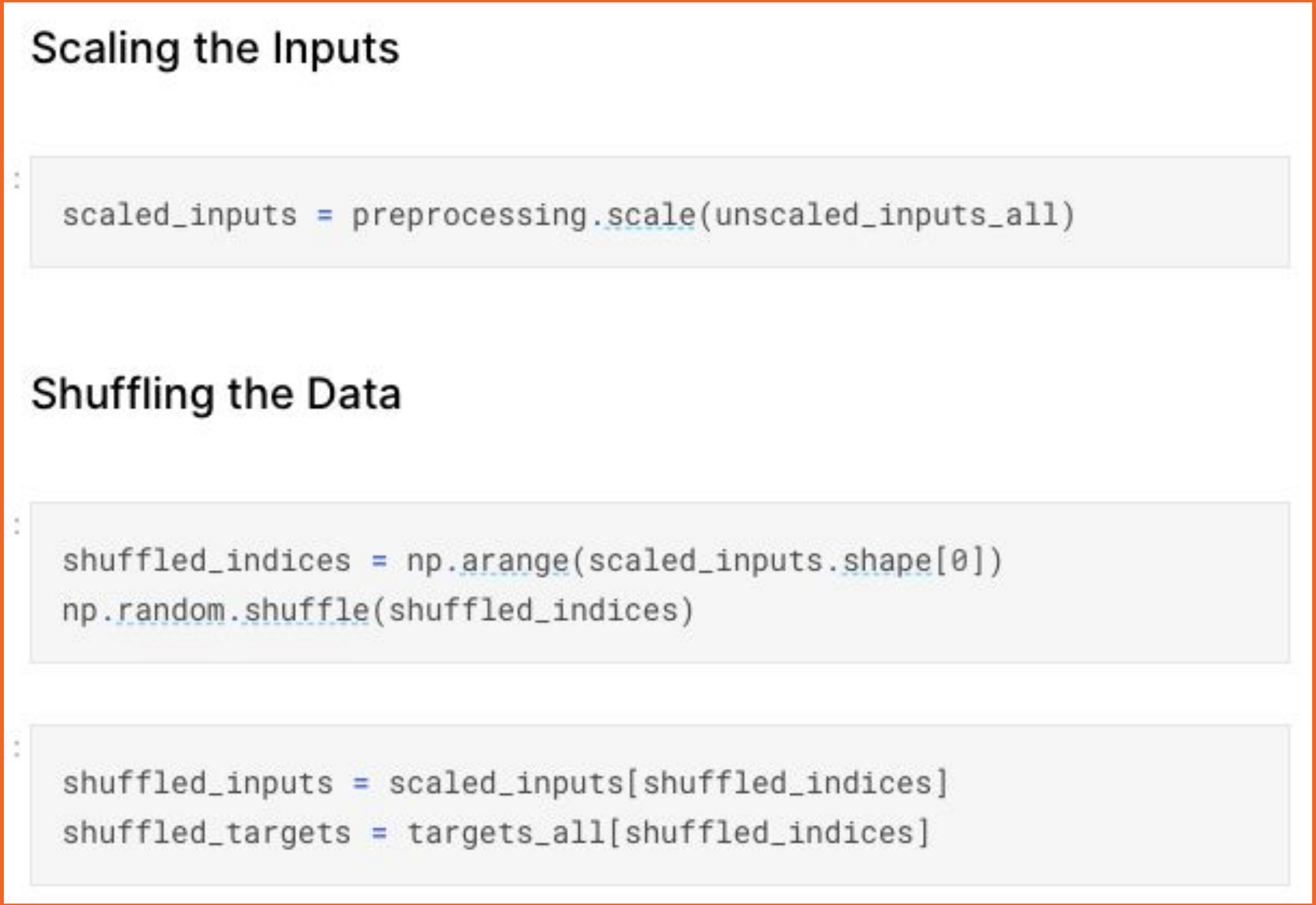
YASH
RANDIVE



```
#import numpy as np
import tensorflow as tf
from sklearn import preprocessing
```

```
unscaled_inputs_all = df_train[['vote_funny', 'vote_cool', 'vote_u
seful','polarity', 'subjectivity']]
targets_all = df_train['rating']
```

YASH RANDIVE



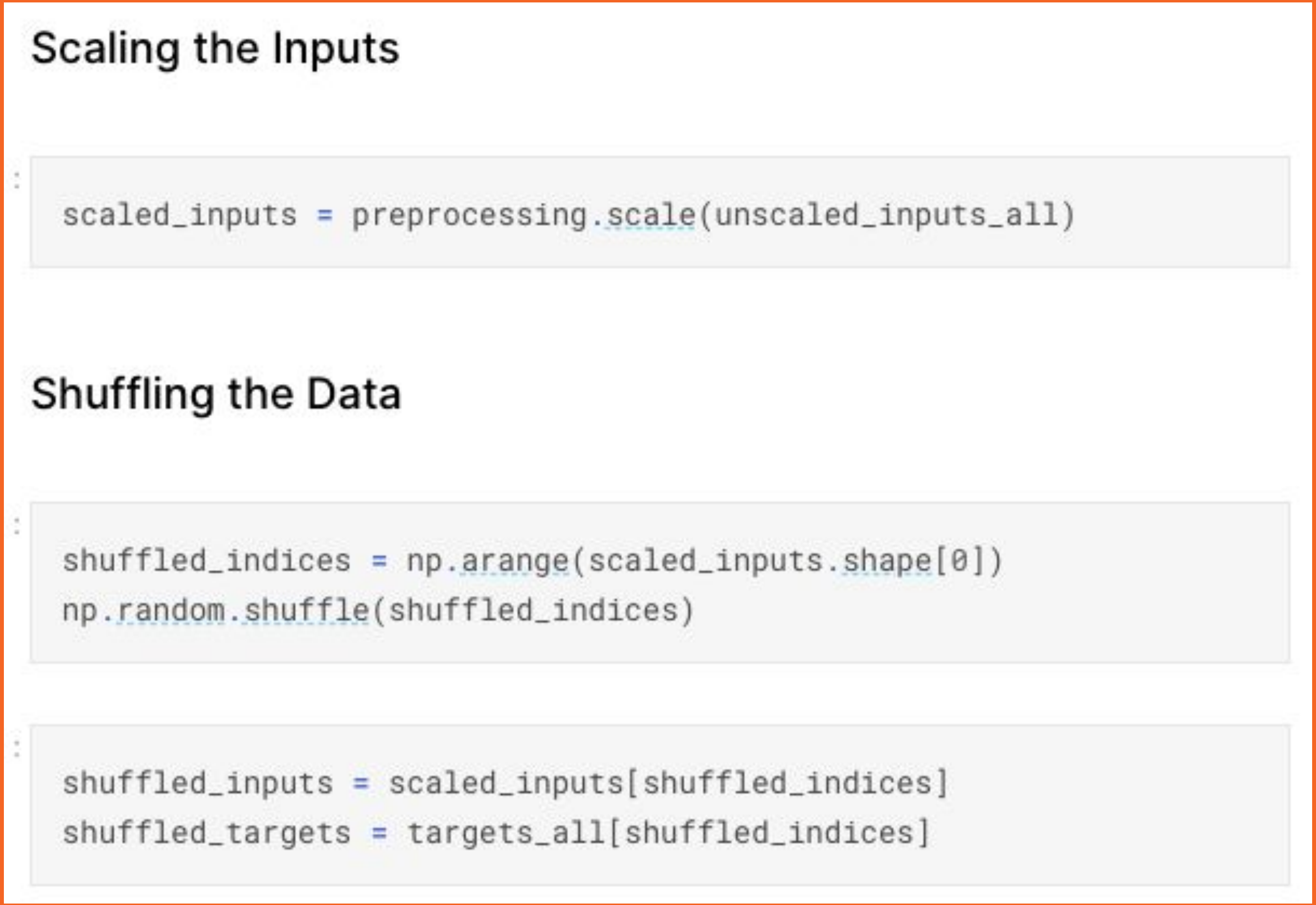
Scaling the Inputs

```
scaled_inputs = preprocessing.scale(unscaled_inputs_all)
```

Shuffling the Data

```
shuffled_indices = np.arange(scaled_inputs.shape[0])
np.random.shuffle(shuffled_indices)
```

```
shuffled_inputs = scaled_inputs[shuffled_indices]
shuffled_targets = targets_all[shuffled_indices]
```



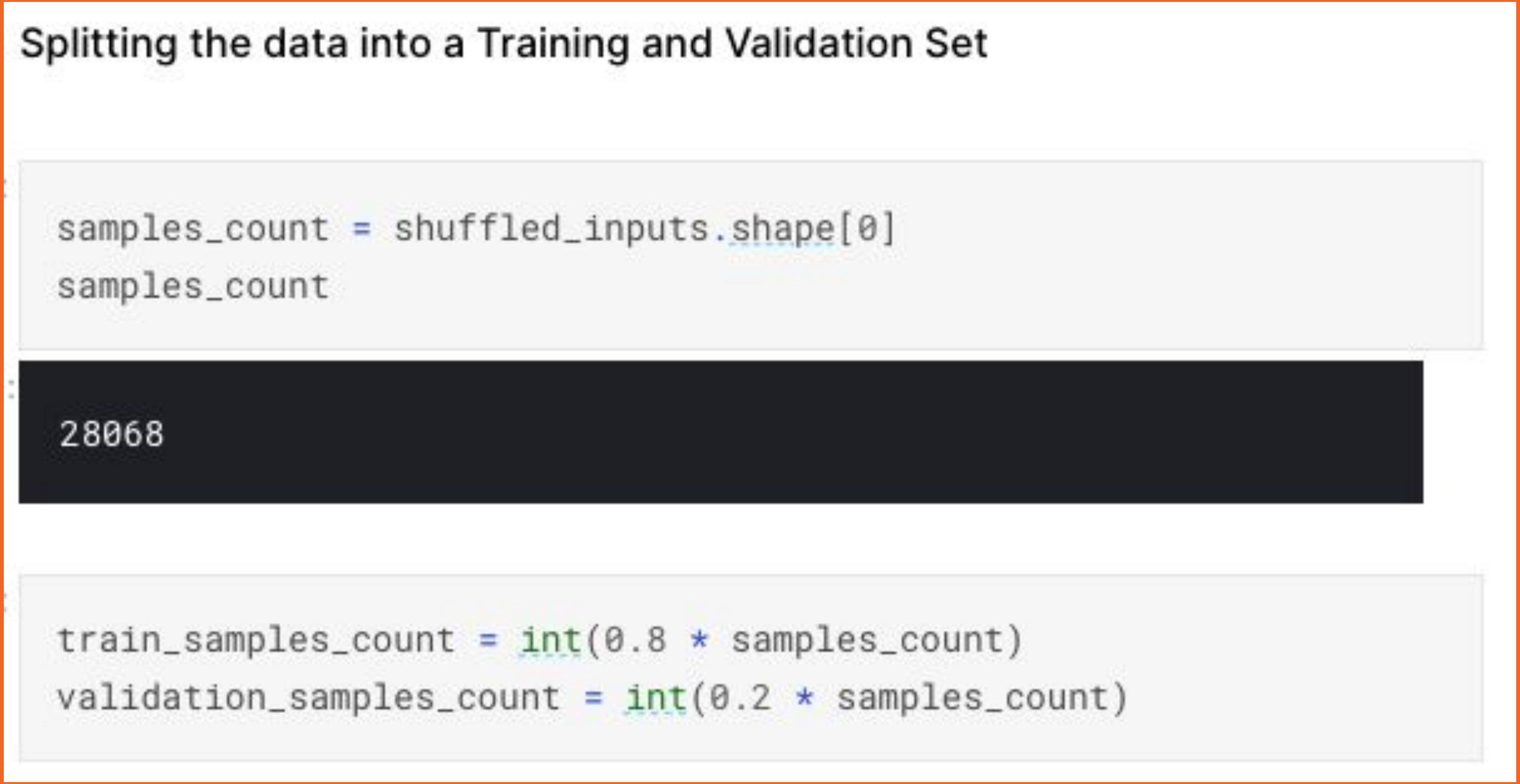
Scaling the Inputs

```
scaled_inputs = preprocessing.scale(unscaled_inputs_all)
```

Shuffling the Data

```
shuffled_indices = np.arange(scaled_inputs.shape[0])
np.random.shuffle(shuffled_indices)
```

```
shuffled_inputs = scaled_inputs[shuffled_indices]
shuffled_targets = targets_all[shuffled_indices]
```



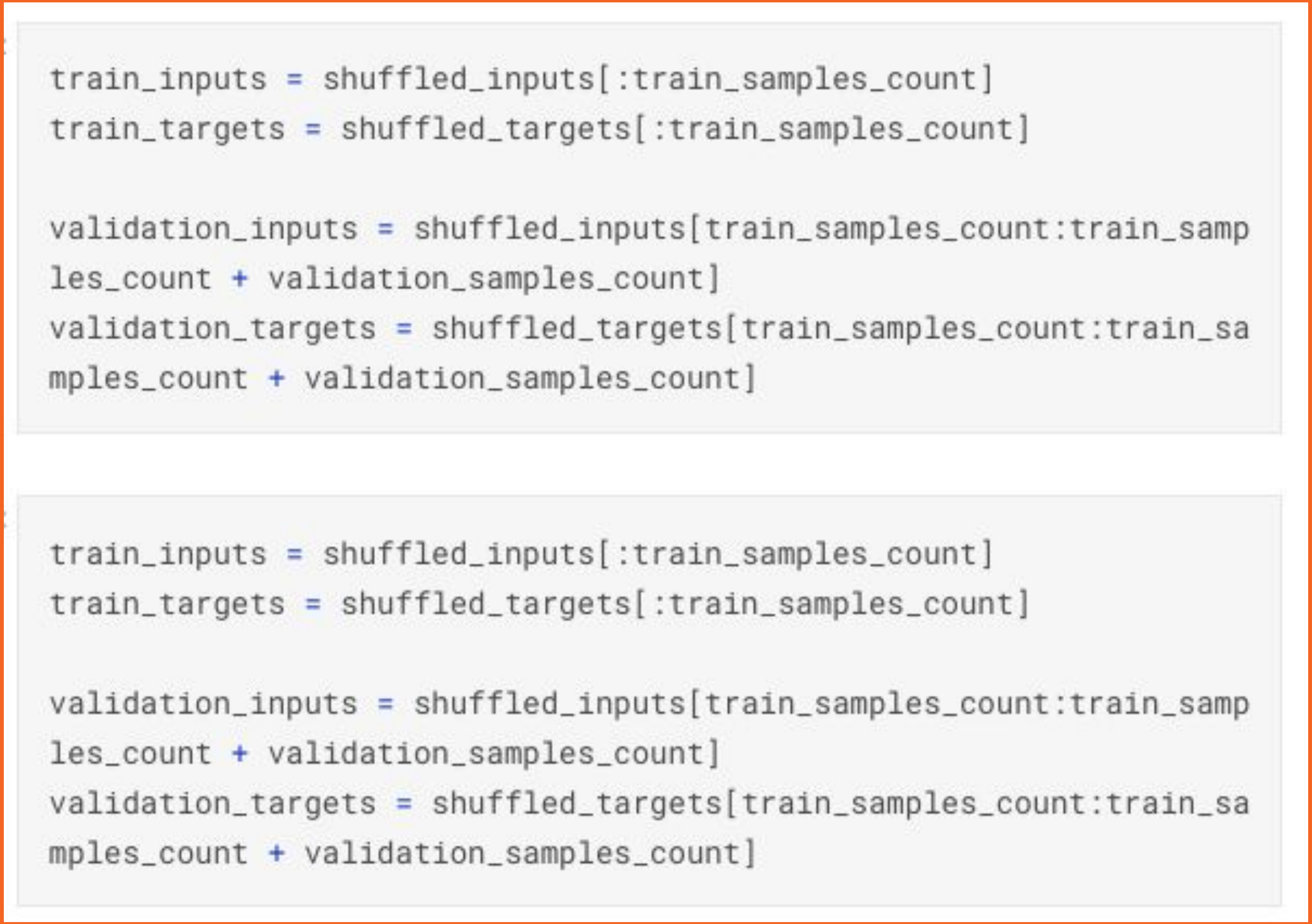
Splitting the data into a Training and Validation Set

```
samples_count = shuffled_inputs.shape[0]
samples_count
```

```
28068
```

```
train_samples_count = int(0.8 * samples_count)
validation_samples_count = int(0.2 * samples_count)
```

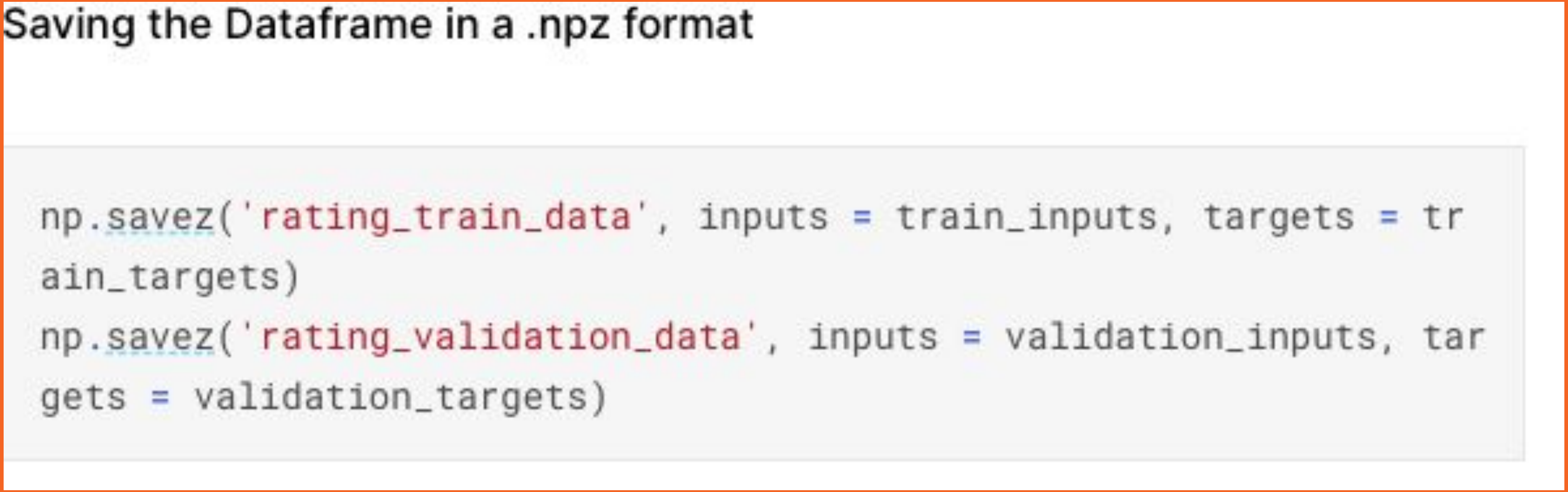
YASH RANDIVE



```
train_inputs = shuffled_inputs[:train_samples_count]
train_targets = shuffled_targets[:train_samples_count]

validation_inputs = shuffled_inputs[train_samples_count:train_samp
les_count + validation_samples_count]
validation_targets = shuffled_targets[train_samples_count:train_sa
mples_count + validation_samples_count]
```

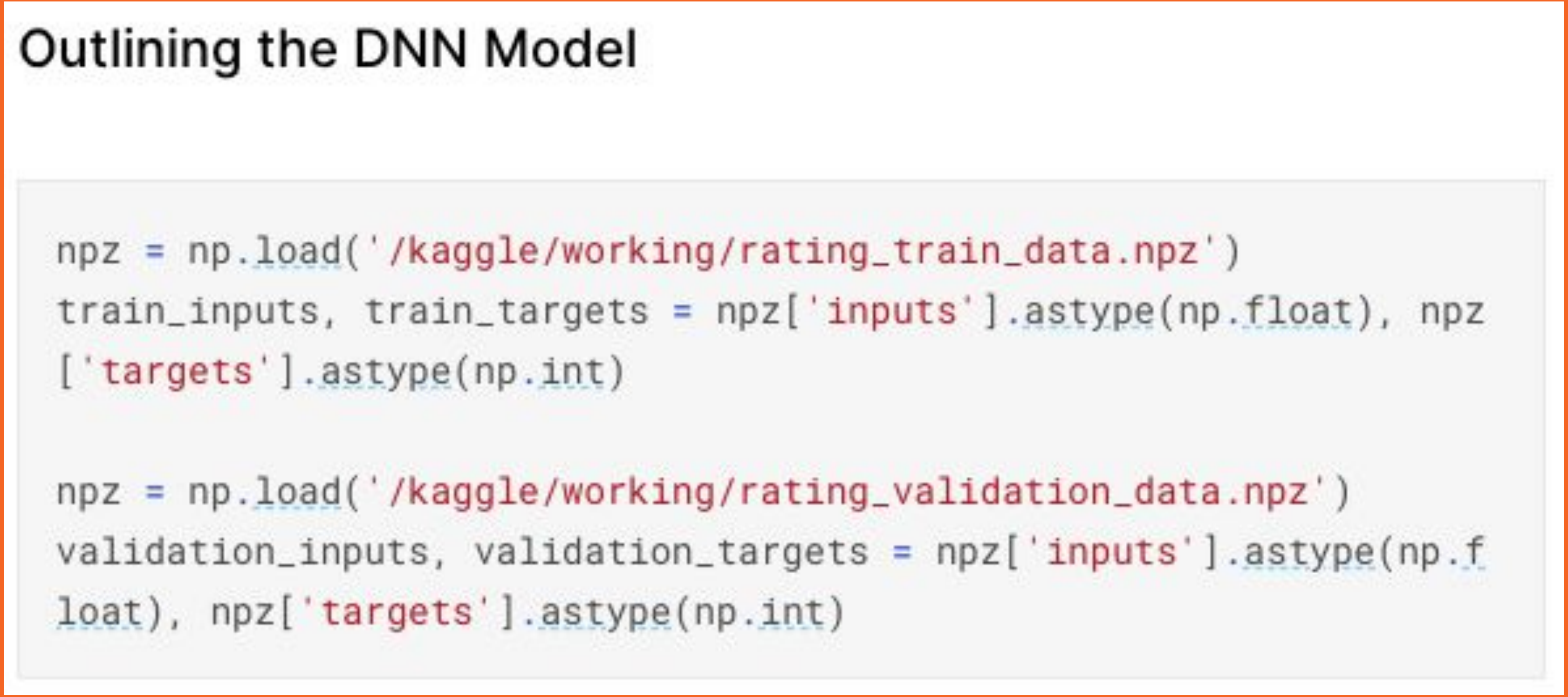
```
train_inputs = shuffled_inputs[:train_samples_count]
train_targets = shuffled_targets[:train_samples_count]

validation_inputs = shuffled_inputs[train_samples_count:train_samp
les_count + validation_samples_count]
validation_targets = shuffled_targets[train_samples_count:train_sa
mples_count + validation_samples_count]
```

YASH
RANDIVE



Saving the Dataframe in a .npz format

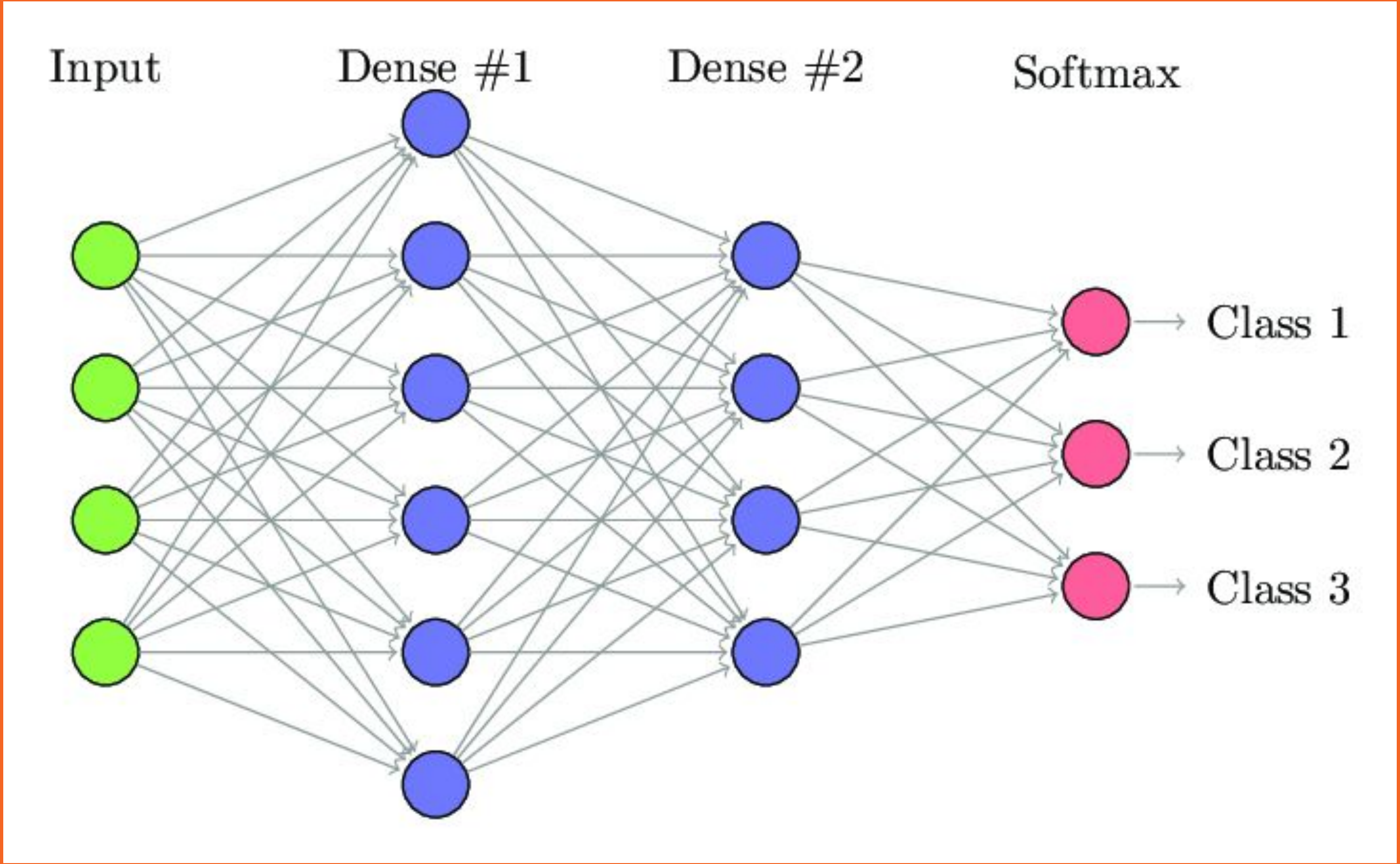
```
np.savez('rating_train_data', inputs = train_inputs, targets = tr
ain_targets)
np.savez('rating_validation_data', inputs = validation_inputs, tar
gets = validation_targets)
```



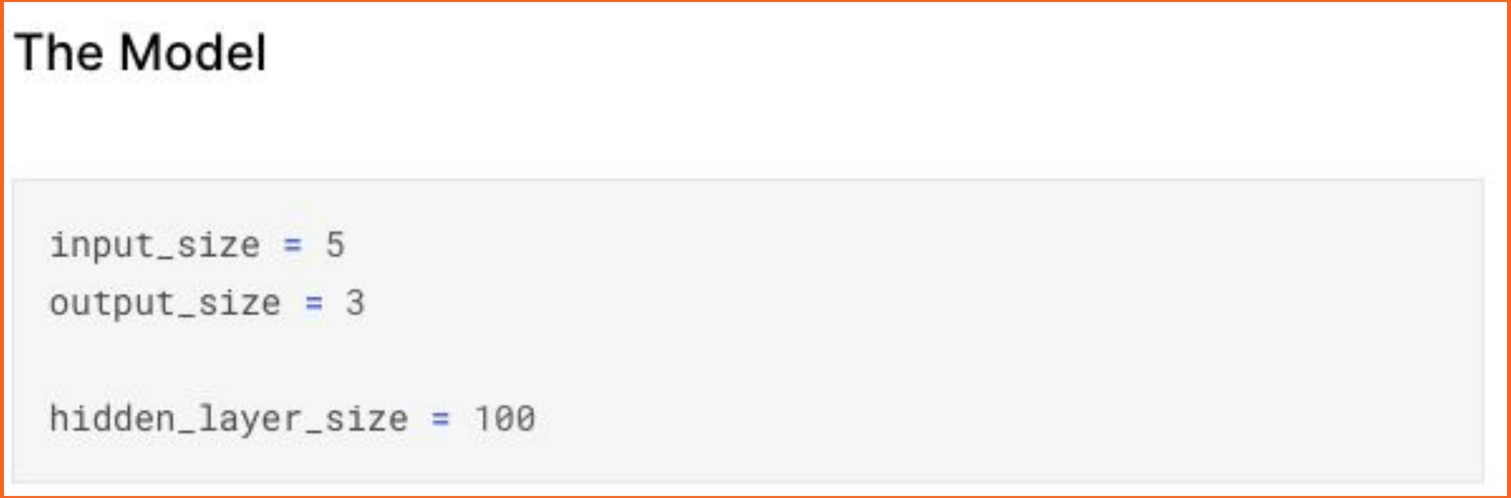
Outlining the DNN Model

```
npz = np.load('/kaggle/working/rating_train_data.npz')
train_inputs, train_targets = npz['inputs'].astype(np.float), npz
['targets'].astype(np.int)

npz = np.load('/kaggle/working/rating_validation_data.npz')
validation_inputs, validation_targets = npz['inputs'].astype(np.f
loat), npz['targets'].astype(np.int)
```

YASH
RANDIVE



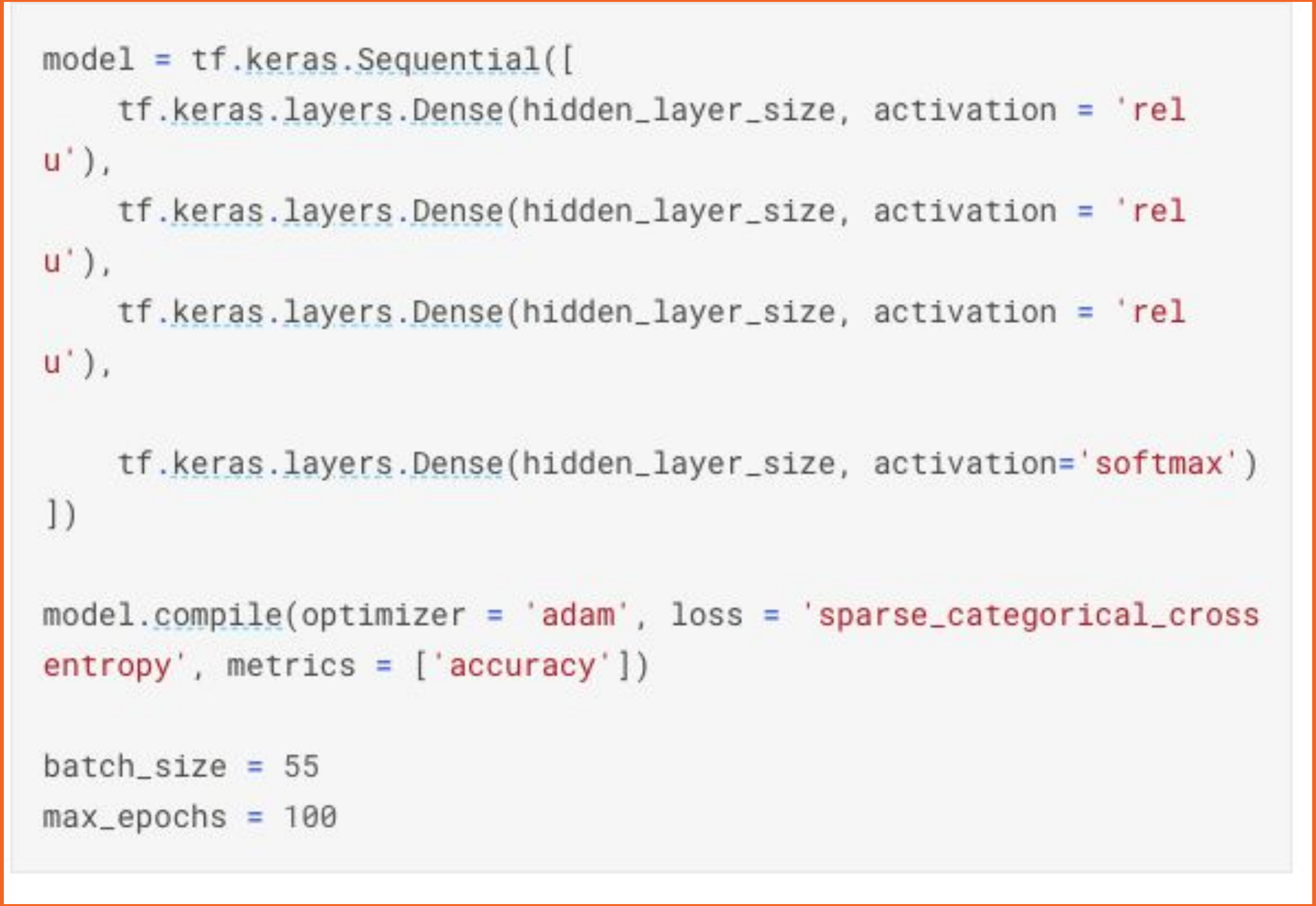
YASH
RANDIVE



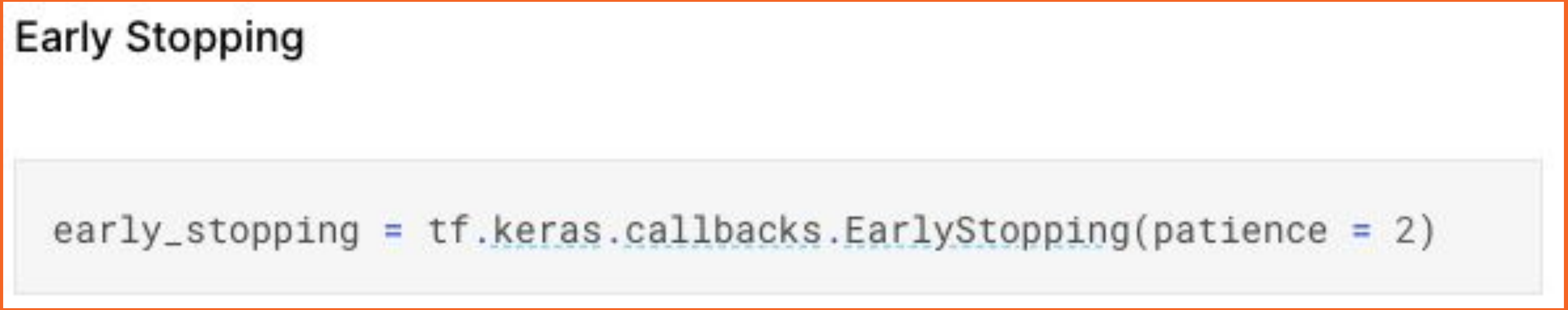
The Model

```
input_size = 5
output_size = 3

hidden_layer_size = 100
```

YASH
RANDIVE



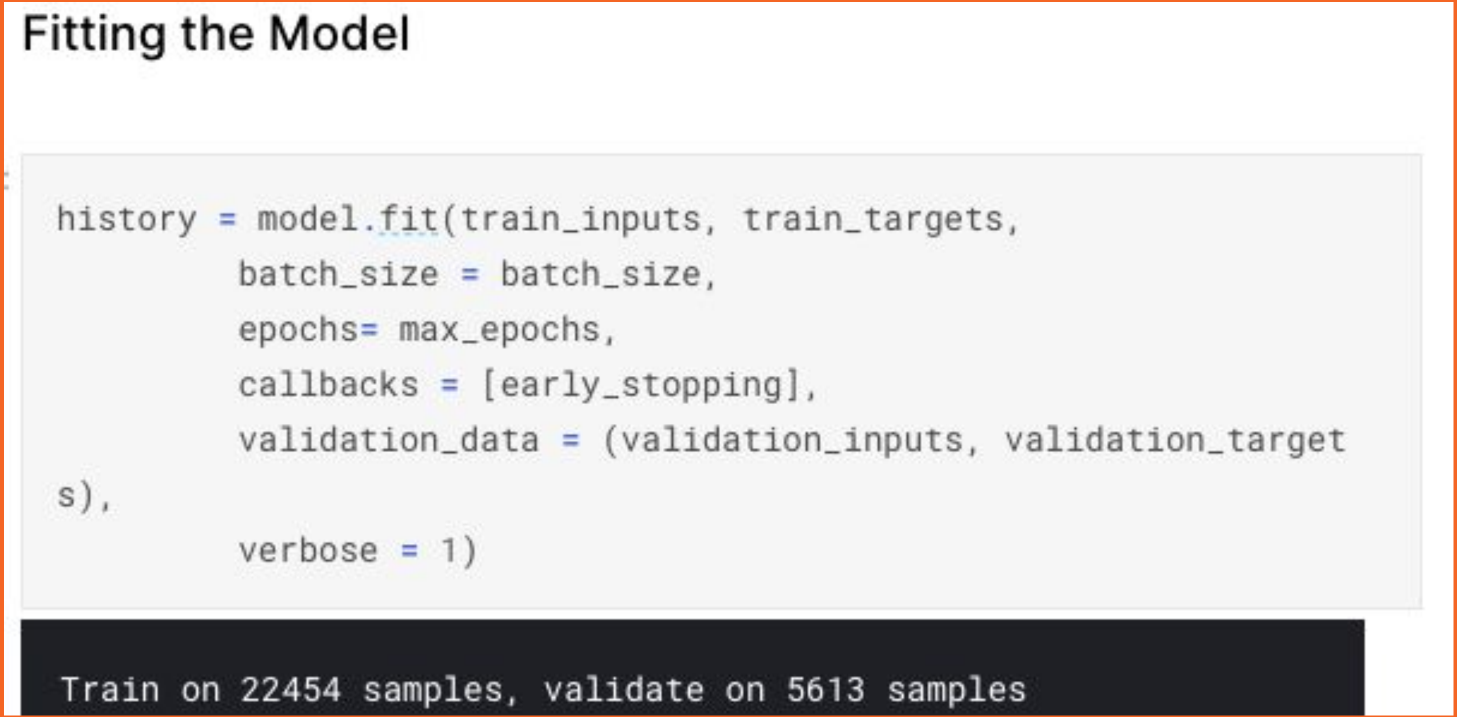
```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(hidden_layer_size, activation = 'rel
u'),
    tf.keras.layers.Dense(hidden_layer_size, activation = 'rel
u'),
    tf.keras.layers.Dense(hidden_layer_size, activation = 'rel
u'),

    tf.keras.layers.Dense(hidden_layer_size, activation='softmax')
])

model.compile(optimizer = 'adam', loss = 'sparse_categorical_cross
entropy', metrics = ['accuracy'])

batch_size = 55
max_epochs = 100
```



Early Stopping

```
early_stopping = tf.keras.callbacks.EarlyStopping(patience = 2)
```

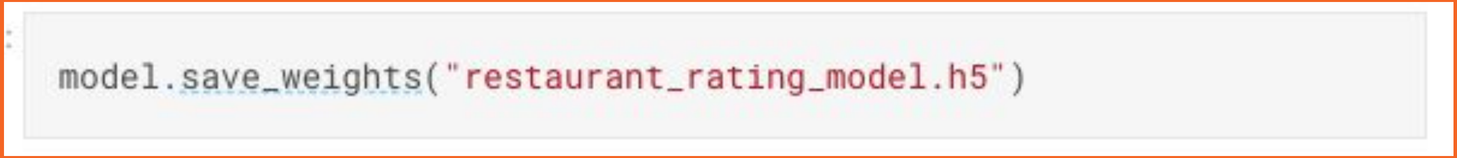
YASH
RANDIVE



Fitting the Model

```
history = model.fit(train_inputs, train_targets,
          batch_size = batch_size,
          epochs= max_epochs,
          callbacks = [early_stopping],
          validation_data = (validation_inputs, validation_target
s),
          verbose = 1)
```

```
Train on 22454 samples, validate on 5613 samples
```

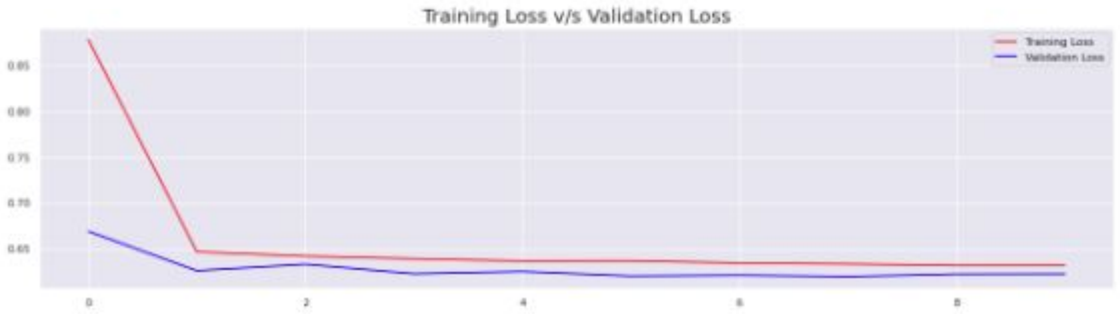


```
model.save_weights("restaurant_rating_model.h5")
```

YASH RANDIVE

Visualising Training

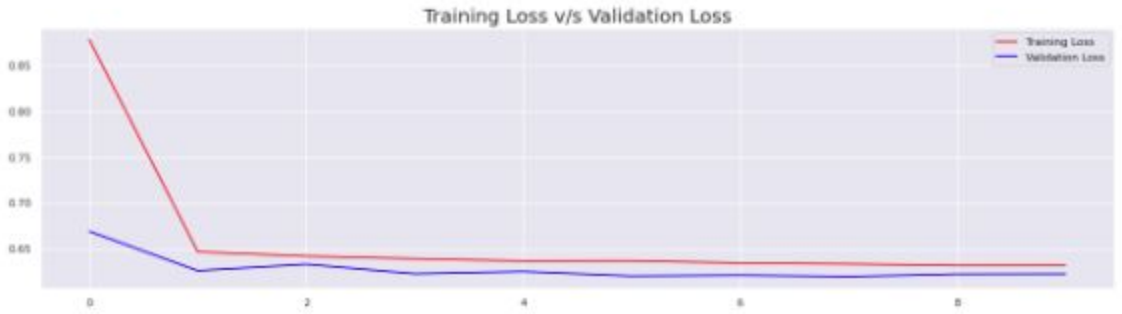
```
plt.figure(figsize = (20,5))
plt.plot(history.history['loss'], color = 'red', label = 'Trainin
g Loss')
plt.plot(history.history['val_loss'], color = 'blue', label = 'Va
lidation Loss')
plt.title('Training Loss v/s Validation Loss', size = 20)
plt.legend()
plt.show()
```



YASH RANDIVE

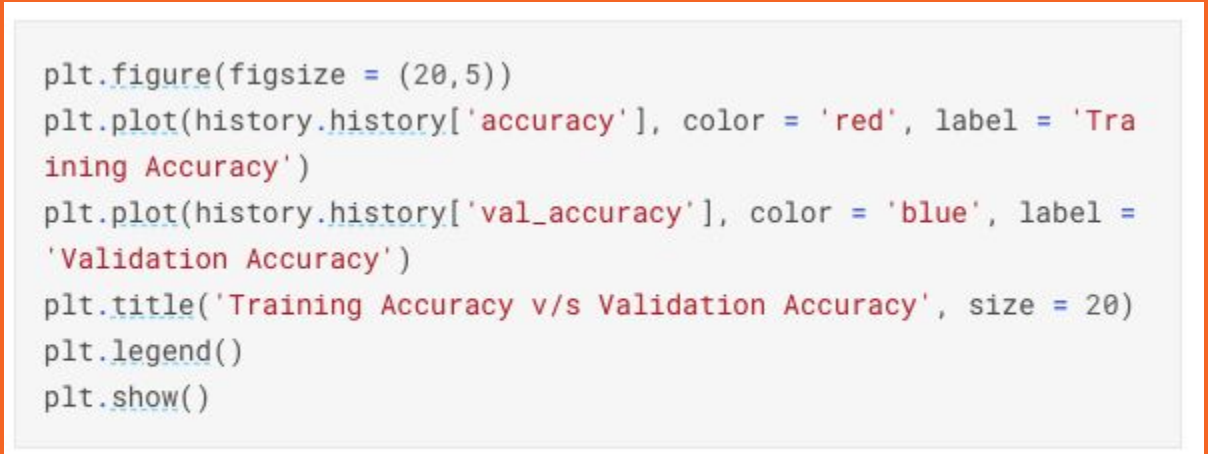
Visualising Training

```
plt.figure(figsize = (20,5))
plt.plot(history.history['loss'], color = 'red', label = 'Training Loss')
plt.plot(history.history['val_loss'], color = 'blue', label = 'Validation Loss')
plt.title('Training Loss v/s Validation Loss', size = 20)
plt.legend()
plt.show()
```

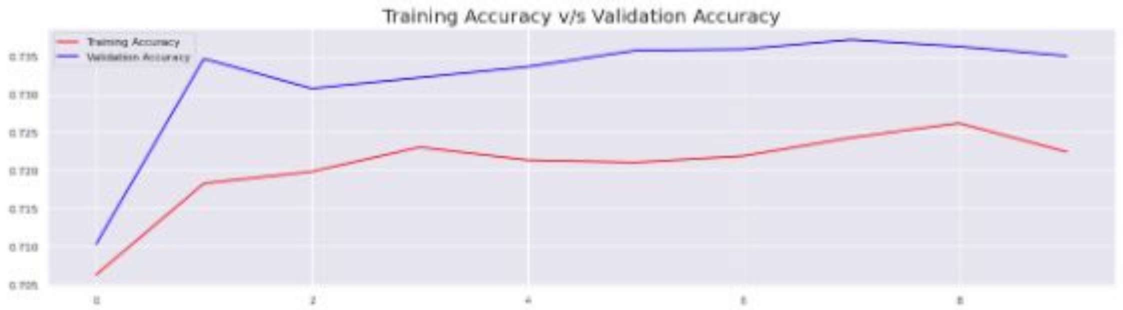


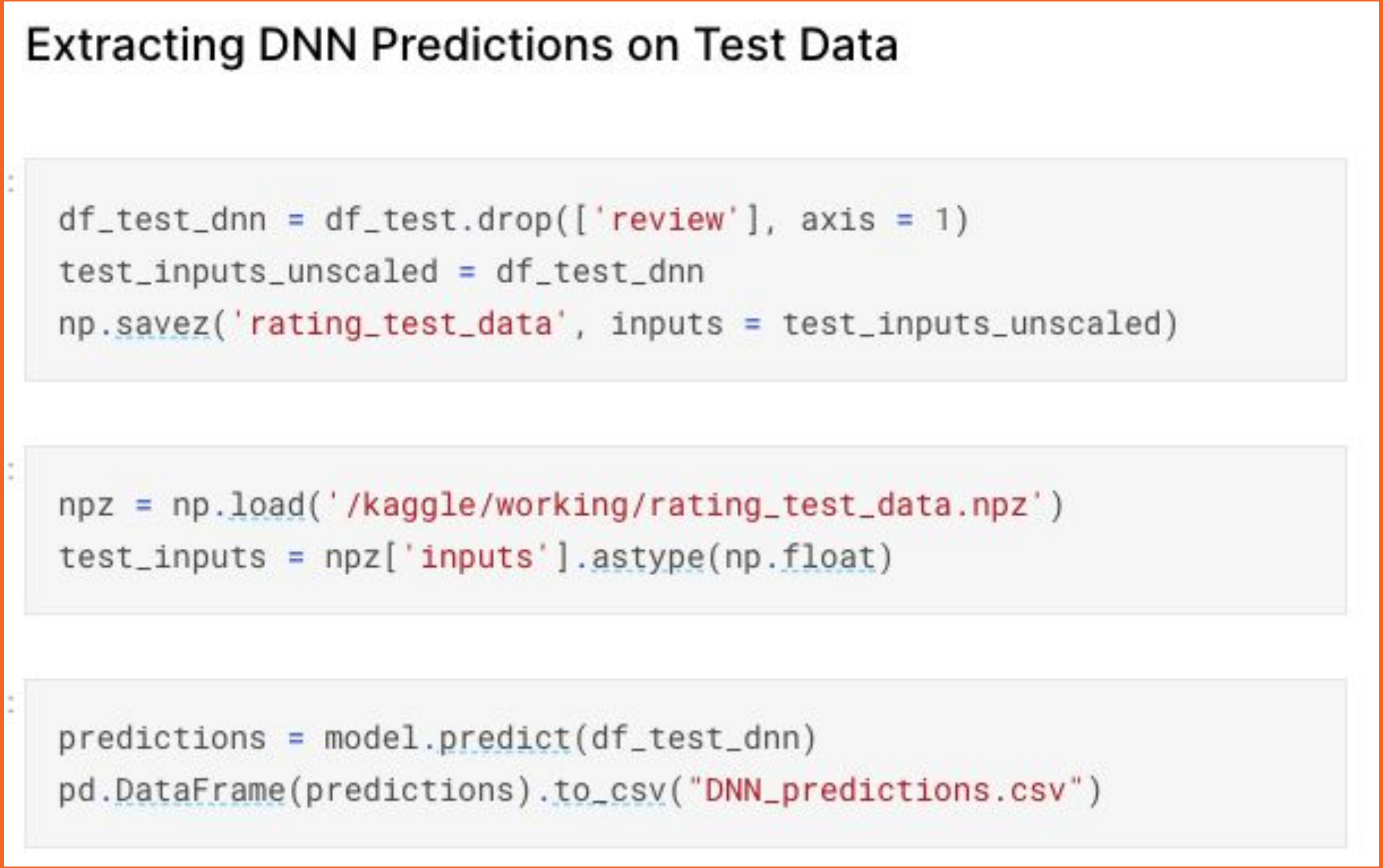
YASH RANDIVE

Visualising Training



```
plt.figure(figsize = (20,5))
plt.plot(history.history['accuracy'], color = 'red', label = 'Training Accuracy')
plt.plot(history.history['val_accuracy'], color = 'blue', label = 'Validation Accuracy')
plt.title('Training Accuracy v/s Validation Accuracy', size = 20)
plt.legend()
plt.show()
```





Extracting DNN Predictions on Test Data

```
df_test_dnn = df_test.drop(['review'], axis = 1)
test_inputs_unscaled = df_test_dnn
np.savez('rating_test_data', inputs = test_inputs_unscaled)
```

```
npz = np.load('/kaggle/working/rating_test_data.npz')
test_inputs = npz['inputs'].astype(np.float)
```

```
predictions = model.predict(df_test_dnn)
pd.DataFrame(predictions).to_csv("DNN_predictions.csv")
```

Congratulations!

You have reached the end of this Project!

For queries or questions in the code reach me at :

yashrandive.datascience@gmail.com

All the Best!