**Task 7: Extension**

**Muhammad Ali, 103960437, COS30018, 06/10/2024, Tutor: Ru Jia.**

**Requirements / Deliverables –**

* Research of the potential approaches for predicting companies’ stock prices/trends.
* Implementation of your selected idea and the evaluation results you have obtained when assessing the performance of the selected idea.

**Result –**

**Summary of research findings –**

1. **Linear Regression**

Linear regression is a basic machine learning algorithm that can be implemented on this dataset. The linear regression model returns an equation determining the relationship between the independent and dependent variables.

The equation for linear regression can be written as:

A black and white text

Description automatically generated

X1, x2,….Xn represent the independent variables while the coefficients O1, O2,….. On represent the weights.

*Disadvantages –*

A green line graph with white background

Description automatically generatedA problem with using regression algorithms in this case is that the model overfits the data from the dataset.

Model prediction – orange.

1. **K-Nearest Neighbours**

kNN finds the similarity between new and old data points, based on the independent variables.

For example:

A graph with numbers and a line

Description automatically generatedConsider the height and age of 11 people (Based on the given features ‘Age’ and ‘Height:

To determine the weight for ID #11. kNN considers the weight of the nearest neighbours of this ID.

The weight of ID #11 is predicted to be the average of its neighbours. If we consider three neighbours (k=3) for now, the weight for ID11 would be:

= (77 + 72 + 60) / 3

= 69.66 kg.

*Implementation –*

*A screenshot of a computer code

Description automatically generated*

Using scaled data –

A close up of text

Description automatically generated

Finding the best parameters –

A close up of a text

Description automatically generated

Fit the model and make predictions –

A close up of words

Description automatically generated

A green line graph with a white background

Description automatically generated

kNN RESULT –

k-Nearest Neighbours is a regression technique as well and like linear regression. This might be because instead of considering the previous values from the point of prediction, the model considers the values from the same date a month ago or the same date/month a year ago.

Due to this *regression models* lead to overfitting.

1. **Prophet**

Prophet is a time series forecasting library that requires no data preprocessing and is extremely simple to implement.

Prophet tries to capture the seasonality in past data and works well when the dataset is large.

*Prophet Results –*

A green line graph with white background

Description automatically generated

Disadvantages –

Unfortunately, stock prices are hardly ever seasonal in nature and are often affected by outside factors like world events, so seasonality is often hard to capture and use for stock predictions. Hence, forecasting techniques like Prophet, ARIMA, and SARIMA would not show good results for stock prediction.

**Implementation of selected idea –**

**Backpropagation –**

Backpropagation is a multilayer feed-forward network. Backpropagation is given a function to model; the neural network modifies the *internal weightings* of input to produce an expected output. The system is trained, where the *error* between the system’s output and the expected output is used to modify its *internal state.*

A backpropagation model can be used for both *regression* and *classification* problems but judging by the results of the regression models I’m using classification to predict.

Implementation –

To implement backpropagation using TensorFlow we need to implement a feedforward neural network, backpropagation will then update the weights during training using the specified *loss function* and *optimiser*.

**Steps –**

1. **Add a function to create a feedforward model.**

In data\_processing.py, we add a new function to create a feedforward model (utilises the Dense, and Flatten functions from TensorFlow library):

def create\_feedforward\_model(input\_shape, hidden\_layers, hidden\_units, activation\_functions, output\_size,  
 loss\_function='mse', optimizer='adam'):  
 *"""  
 Creates a feedforward neural network model.  
  
 Parameters:  
 - input\_shape (tuple): Shape of the input data (timesteps, features).  
 - hidden\_layers (int): Number of hidden layers.  
 - hidden\_units (list of int): Number of units in each hidden layer.  
 - activation\_functions (list of str): Activation functions for each hidden layer.  
 - output\_size (int): Number of units in the output layer.  
 - loss\_function (str): Loss function to use for training.  
 - optimizer (str): Optimizer to use for training.  
  
 Returns:  
 - model (Sequential): Compiled Keras feedforward model.  
 """* model = Sequential()  
 model.add(Flatten(input\_shape=input\_shape))  
  
 for i in range(hidden\_layers):  
 model.add(Dense(hidden\_units[i], activation=activation\_functions[i]))  
  
 model.add(Dense(output\_size)) # Output layer  
 model.compile(optimizer=optimizer, loss=loss\_function)  
  
 return model

1. **Integrate FF model into main program.**

The feedforward model is implemented into the previous program that uses LSTM and SARIMA in an ensemble approach to provide predictions.

1. **The data input shape is adjusted for the backpropagation model:**

input\_shape\_ff = (PREDICTION\_DAYS, X\_train\_scaled.shape[2])  
hidden\_layers = 3  
hidden\_units = [100, 50, 25]  
activation\_functions\_ff = ['relu', 'relu', 'relu']

1. **I create and compile the backpropagation model; I save its best model output after model training:**
2. # Create and compile the backpropagation model  
   backpropagation\_model = create\_backpropagation\_model(  
    input\_shape=input\_shape\_ff,  
    hidden\_layers=hidden\_layers,  
    hidden\_units=hidden\_units,  
    activation\_functions=activation\_functions\_ff,  
    output\_size=FUTURE\_STEPS,  
    loss\_function='huber',  
    optimizer='adam'  
   )
3. # Define early stopping and checkpoint for the backpropagation model  
   checkpoint\_ff = ModelCheckpoint('best\_ff\_model.h5', save\_best\_only=True, monitor='val\_loss')  
     
   # Train the backpropagation model  
   history\_ff = backpropagation\_model.fit(  
    X\_train\_scaled, y\_train\_scaled,  
    epochs=100, batch\_size=32, verbose=1,  
    validation\_split=0.2, callbacks=[early\_stopping, checkpoint\_ff]
4. # Predict with the backpropagation model  
   backpropagation\_model.load\_weights('best\_ff\_model.h5')  
   predicted\_ff = backpropagation\_model.predict(X\_test\_scaled)

I implemented the feedforward network and adjusted the weights according to each model’s input to the predictions output.

# Combine DL, SARIMA, and Feedforward predictions using weighted average  
weight\_dl = 0.5  
weight\_sarima = 0.2  
weight\_ff = 0.3

The LSTM model has a 50% weighting to the output, the SARIMA model has 20% due to its seasonal predictive output, and the new backpropagation model has a 30% weight to the prediction output.

**Results –**

Backpropagation provides a positive result to the stock prediction using the stock price, due to its feedforward network whose output is backpropagated into the model by accounting for its error/loss, therefore improving on the model’s previous predictions and outputting an improved result. Backpropagation combined with the LSTM/SARIMA ensemble method produces a complete ensemble approach for stock prediction.

A graph showing a price prediction

Description automatically generated