Task 6 Report

NACA HITCHMAN - 103072170

ARIMA & Ensemble

ARIMA is another ML model designed for time series prediction, making it applicable for this stock prediction program. It's implementation and usage however does differ to that of an LTSM model.

The first step is to simply import the ARIMA library into the program. This also involved installing the statsmodels package through pip, and including said package in the requirements.txt

```
from statsmodels.tsa.arima.model import ARIMA
```

My first attempt at implementing involved creating a new ARIMA model with the test data, and trying to run the forecast() method on with, passing as the number of steps the length of the test data. This method proved to be unsuccessful despite my efforts to determine why and attempts to fix it. Eventually, I determined it appears forecast() is more reliable on smaller step counts, and created a new strategy to create the prediction.

To be able to predict on the entire test set, I needed to create a loop over the length of a test data, and essentially create and fit a new ARIMA model each loop. Within each iteration, the model would be created using the existing train data, after appending the actual value of test data from the prior iteration. The forecast would then just be run for the next day, before being added to the final prediction array and moving to the next iteration of the loop.

```
test_data = data['test_data'][prediction_column].values[-
len(predicted_close_prices):]
history = [x for x in data['train_data'][prediction_column].values]
predictions = list()
for t in range(len(test_data)):
        arima_model = ARIMA(history, order=(5,1,0))
        model_fit = arima_model.fit()
        output = model_fit.forecast()
        forcast = output[0]
        predictions.append(forcast)
        obs = test_data[t]
        history.append(obs)
        print('%f/%f, predicted=%f, expected=%f' % (t,len(test_data), forcast, obs))
```

To then create an ensemble prediction using the average between the ARIMA and LSTM results, firstly the arima results needed to be inverse scaled to match the actual results, and then reshaped to the same shape as the LSTM prediction. Then a simple averaging calculation is run on the two prediction arrays to create the ensemble prediction.

```
arima_predictions_scaled =
data["column_scaler"][prediction_column].inverse_transform(np.array(predictions
).reshape(-1,1)).reshape(-1)
ensemble_prediction = (predicted_close_prices + arima_predictions_scaled) / 2
```

Getting the next k days prediction was also simple, by creating another ARIMA model, fitting it, then using forecast passing the number of steps we want to predict. The ensemble prediction can also be calculated the same way as before.

```
arima_model = ARIMA(data['train_data'][prediction_column], order=(5,1,0))
model_fit = arima_model.fit()

arima_prediction = model_fit.forecast(steps=N_STEPS)
arima_prediction =
data["column_scaler"][prediction_column].inverse_transform(np.array(arima_predi
ction).reshape(-1,1)).ravel()
ensemble_prediction = (prediction + arima_prediction) / 2

for i, price in enumerate(prediction):
    print(f"LTSM Prediction for day {i+1}: {price}")
for i, price in enumerate(arima_prediction):
    print(f"ARIMA Prediction for day {i+1}: {price}")
for i, price in enumerate(ensemble_prediction):
    print(f"Ensemble Prediction for day {i+1}: {price}")
```

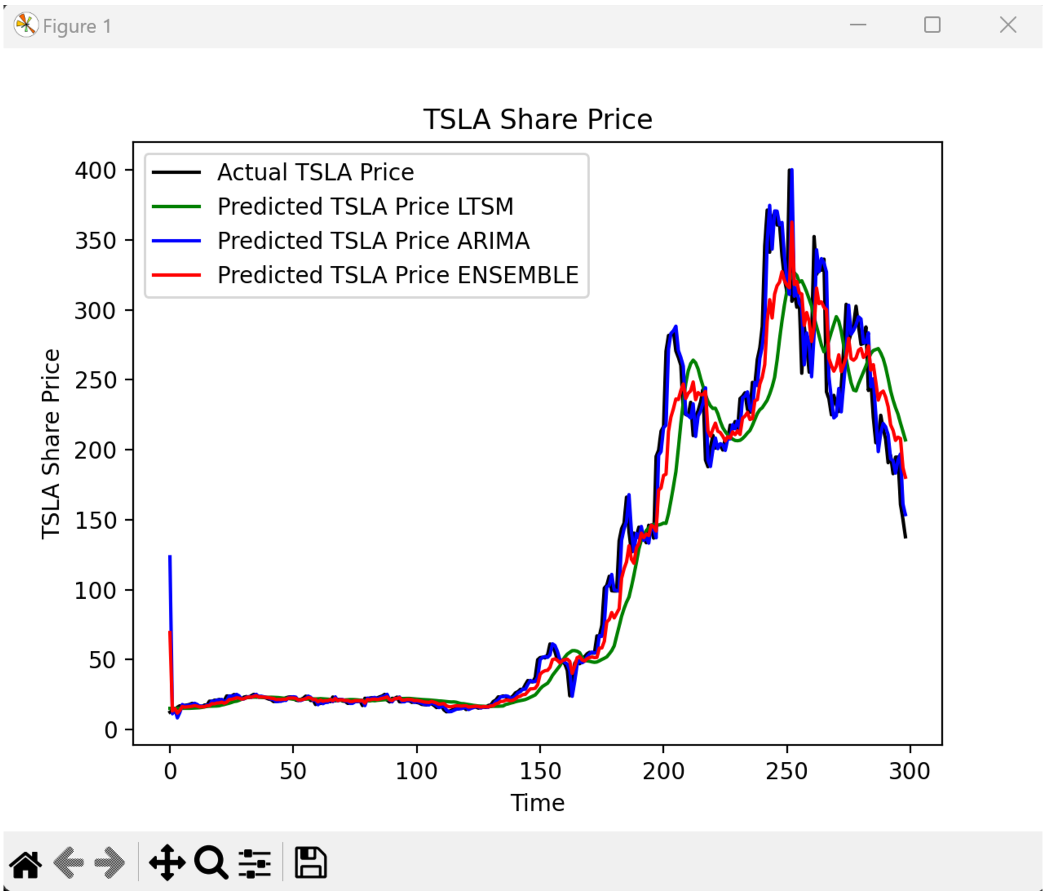
Random Forrest Ensemble

To further test ensemble predictions with other model configurations, I attempted to add in Random Forrest model and make predictions with that. The implementation involved creating a new RF model, in this case passing 300 estimators (or ‘trees), and a random state of 42. The data was then shaped to fit the model fit input shape before the model is fit and predicted on. The ensemble prediction is then calculated in the same way as earlier.

```
rf = RandomForestRegressor(n_estimators=300, random_state=42)
X_train_flattened = data["X_train"][:, :,
closing_price_index].reshape(data["X_train"].shape[0], -1)

y_train_reshaped = data["y_train"][:, :, closing_price_index]
rf.fit(X_train_flattened, y_train_reshaped)

X_test_2d = data["X_test"].reshape((data["X_test"].shape[0], -1))
rf_predictions = rf.predict(data["X_test"][:, :, closing_price_index])

rf_predictions =
data["column_scaler"][prediction_column].inverse_transform(np.array(rf_prediction
s).reshape(-1,1)).reshape(-1)

rf_predictions = rf_predictions[-len(predicted_close_prices):]

ensemble_prediction = (predicted_close_prices + rf_predictions) / 2
```

Results

As can be seen from the below results, the ARIMA prediction is quite accurate, but this may also be due to the way the prediction is created, adding the true test value to each iteration rather than only building off the previous predicted value. Due to the LSTM result being not very accurate, it also means the subsequent ensemble result is not as accurate. This can also be seen in the next k days prediction results.



```
LTSM Prediction for day 1: 240.41233825683594
LTSM Prediction for day 2: 235.5443878173828
LTSM Prediction for day 3: 229.12179565429688
LTSM Prediction for day 4: 220.3579864501953
LTSM Prediction for day 5: 207.03627014160156
ARIMA Prediction for day 1: 123.43450947465706
ARIMA Prediction for day 2: 124.15168619563438
ARIMA Prediction for day 3: 124.25037227507583
ARIMA Prediction for day 4: 124.16050611888188
ARIMA Prediction for day 5: 124.18167944751954
Ensemble Prediction for day 1: 181.92342386574649
Ensemble Prediction for day 2: 179.8480370065086
Ensemble Prediction for day 3: 176.68608396468636
Ensemble Prediction for day 4: 172.2592462845386
Ensemble Prediction for day 5: 165.60897479456054
```

As for the random forrest results, it does I've likely done something wrong with either the model or the data preparation, as the output is completely incorrect. It does still demonstrate that an ensemble average result can be determined from different configurations and models.

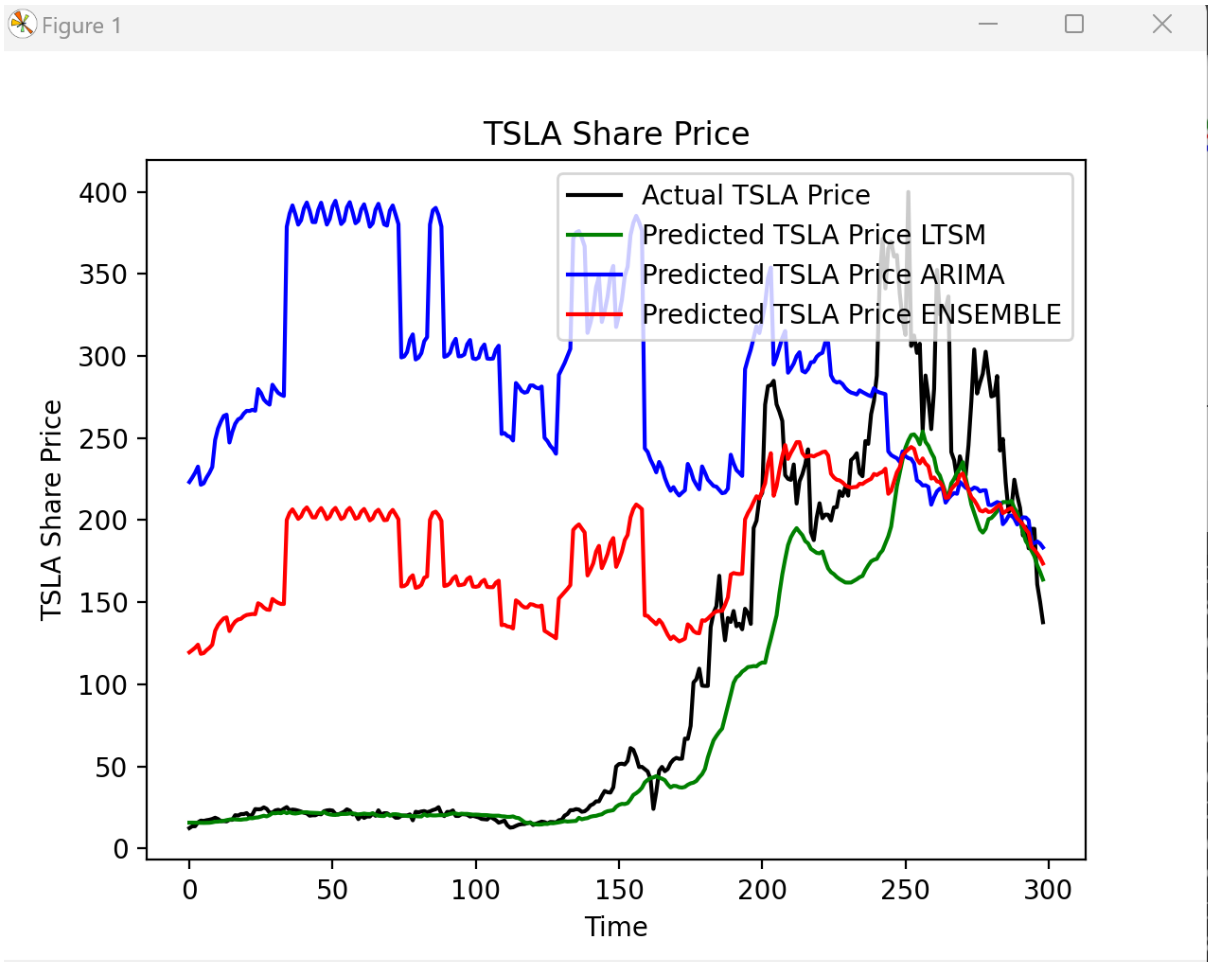


Figure 1
TSLA Share Price
Actual TSLA Price
Predicted TSLA Price LTSM
Predicted TSLA Price ARIMA
Predicted TSLA Price ENSEMBLE
TSLA Share Price
0
50
100
150
200
250
300
350
400
0
50
100
150
200
250
300
Time