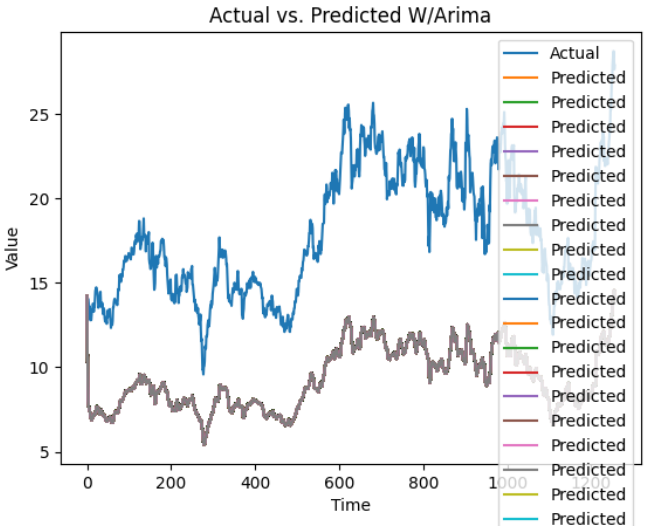
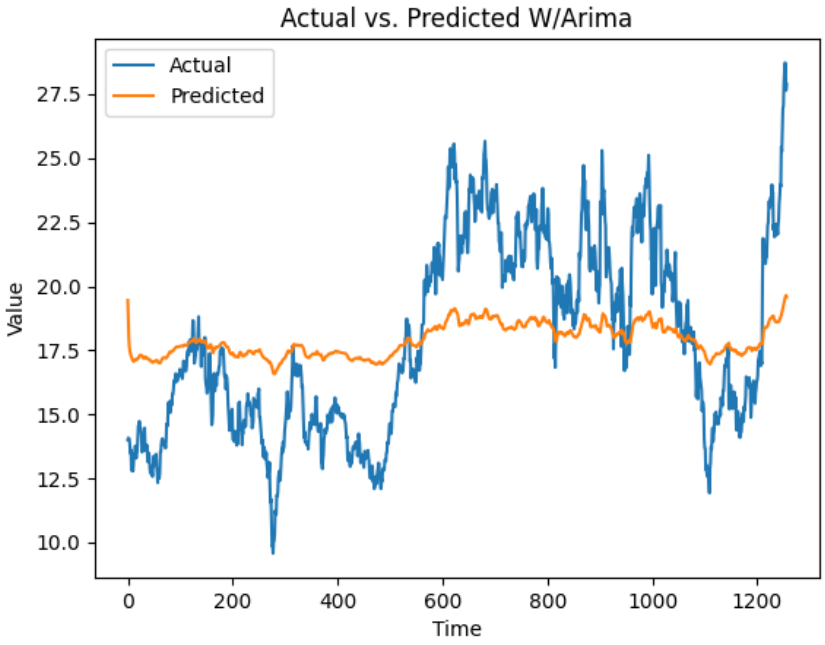
Created arima function and used a simple average to combine it with an existing ai prediction.

```python
def ensemble_arima_lstm(train_data, test_data, columns, p_values, d_values, q_values, lstm_prediction):
    arima_predictions = []

    for column in columns:
        if column == "Close":
            selected_variable = train_data[:, columns.index(column)]
            # best_order = find_best_arima_order(selected_variable, p_values, d_values, q_values) #comment 3 lines to not run optimiser
            # print(f"Best order is: {best_order}") #comment 3 lines to not run optimiser
            # arima_model = ARIMA(selected_variable, order=best_order) #comment 3 lines to not run optimiser
            arima_model = ARIMA(selected_variable, order=[1,0,1]) # un comment line to not run optimiser
            arima_model_fit = arima_model.fit()
            arima_preds = arima_model_fit.forecast(steps=len(test_data))
            arima_predictions.append(list(arima_preds))

    print("model made")

    ensemble_predictions = np.array(arima_predictions)
    ensemble_predictions = np.transpose(ensemble_predictions)

    # Combine LSTM and ARIMA predictions by taking the average element-wise
    final_ensemble = (lstm_prediction + ensemble_predictions) / 2


    return final_ensemble, ensemble_predictions
```

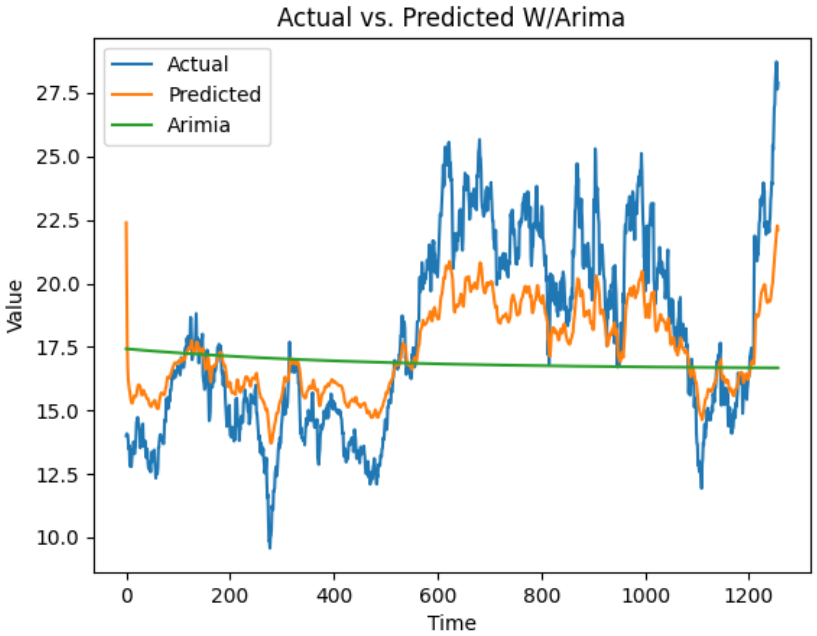
The results I originally tried to make the model use all data columns but later realized I only needed the Close column. There where a few issues I encountered




The first of which was results seeming super low and the results being a 2d array instead of a 1d array, this was a result of me not unscaling the prediction and the arima prediction and lstm predictions being opposing shape (1:x and x:1) so when I added the became x:x. this was fixed by transposing one of the predictions.




However this then lead to my prediciton still being super off which ended up being an error with my how I combined the results. And after fixing that I ended up with this prediciton




This prediction still has several errors which could be a number of factors, the data I'm using could be inaccurate/unsuitable for arima. The prediction being a very slight curve down which doesn't match the expected result which whould be some sort of incline.

in an attempt to fix it I used a method of testing hyper parameters by finding the lowers AIC but this had negligible effects s

```python
def evaluate_order(p, d, q, train_data):
    try:
        print(f"testing: {p} {d} {q}")
        arima_model = ARIMA(train_data, order=(p, d, q))
        arima_model_fit = arima_model.fit()
        aic = arima_model_fit.aic
        return p, d, q, aic
    except Exception as e:
        return p, d, q, float("inf")

def find_best_arima_order(train_data, p_values, d_values, q_values):
    best_aic = float("inf")
    best_order = None

    for p in p_values:
        for d in d_values:
            for q in q_values:
                p, d, q, aic = evaluate_order(p, d, q, train_data)
                if aic < best_aic:
                    best_aic = aic
                    best_order = (p, d, q)
                    print(f"best order: {best_order}")

    if best_order is None:
        best_order = (1, 0, 1)

    return best_order
```

could however just need to test more parameters. (a small search of 200 combinations took close to an hour on my machine)