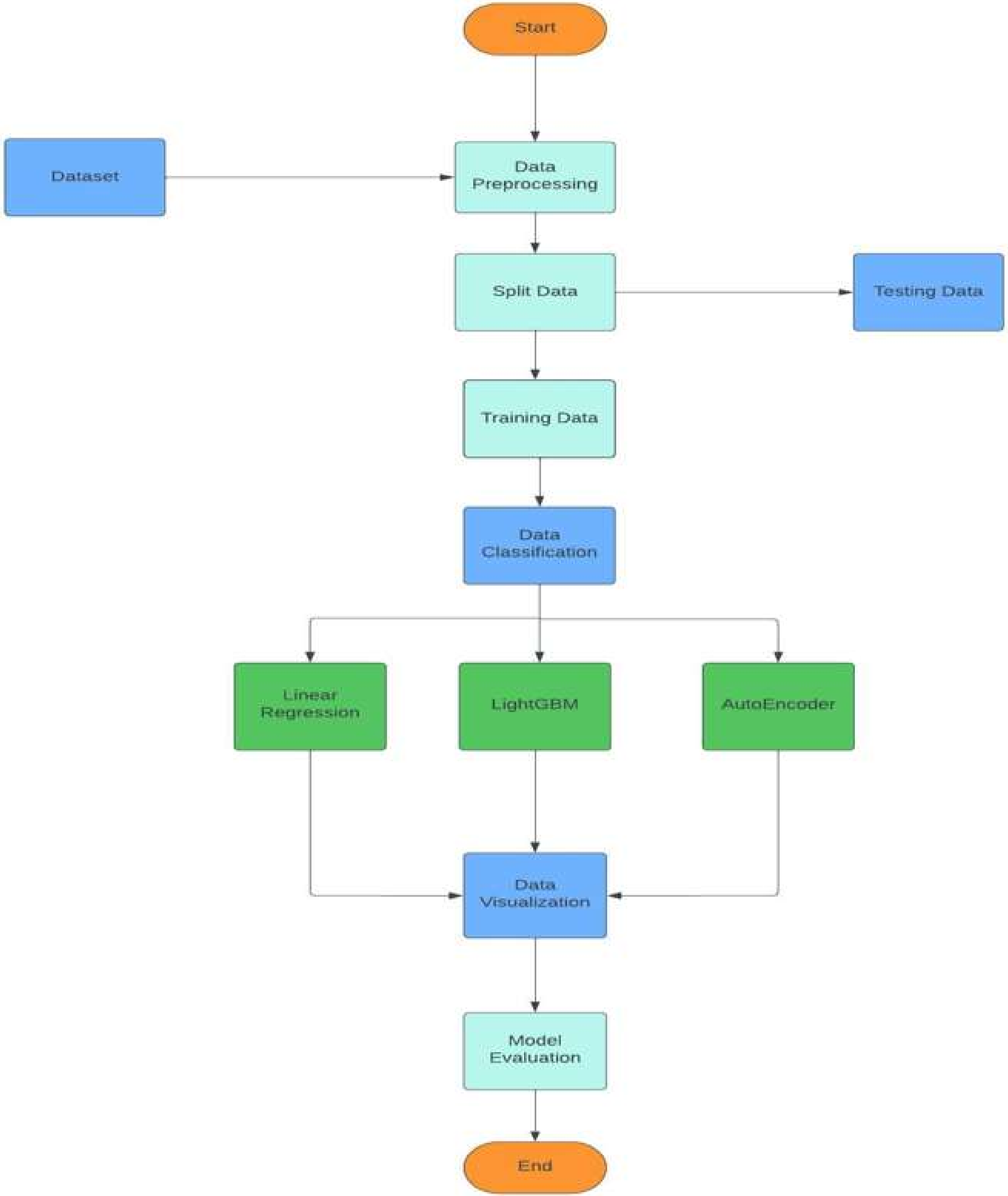

Start
Dataset
Data Preprocessing
Split Data
Testing Data
Training Data
Data Classification
Linear Regression
LightGBM
AutoEncoder
Data Visualization
Model Evaluation
End

```
#split data into training and test data
data_training=pd.DataFrame(df["Close"][0:int(len(df)*0.70)])
data_testing=pd.DataFrame(df["Close"][int(len(df)*0.70):int(len(df))])
print(data_training.shape)
print(data_testing.shape)

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
data_training_array=scaler.fit_transform(data_training)

model=load_model("Keras_model.h5")


past_100_days=data_training.tail(100)
final_df=past_100_days.append(data_testing,ignore_index=True)
input_data=scaler.fit_transform(final_df)

x_test=[]
y_test=[]

for i in range(100,input_data.shape[0]):
    x_test.append(input_data[i-100:i])
    y_test.append(input_data[i,0])

x_test,y_test=np.array(x_test),np.array(y_test)
y_predicted=model.predict(x_test)
scaler.scale_
scale_factor=1/0.02099517
y_predicted=y_predicted*scale_factor
y_test=y_test*scale_factor

st.subheader("Prediction VS Original")
fig2=plt.figure(figsize=(12,6))
plt.plot(y_test,"b",label="Original Price")
plt.plot(y_predicted,"r",label="Predicted Price")
```

Code

Python 3 (ipykernel)

```python
print(data_testing.shape)

from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler(feature_range=(0,1))
data_training_array=scaler.fit_transform(data_training)

model=load_model("Keras_model.h5")


past_100_days=data_training.tail(100)
final_df=past_100_days.append(data_testing,ignore_index=True)
input_data=scaler.fit_transform(final_df)

x_test=[]
y_test=[]

for i in range(100,input_data.shape[0]):
    x_test.append(input_data[i-100:i])
    y_test.append(input_data[i,0])

x_test,y_test=np.array(x_test),np.array(y_test)
y_predicted=model.predict(x_test)
scaler.scale_
scale_factor=1/0.02099517
y_predicted=y_predicted*scale_factor
y_test=y_test*scale_factor

st.subheader("Prediction VS Original")
fig2=plt.figure(figsize=(12,6))
plt.plot(y_test,"b",label="Original Price")
plt.plot(y_predicted,"r",label="Predicted Price")
plt.xlabel("Time")
plt.ylabel("Price")
plt.legend()
st.pyplot(fig2
```

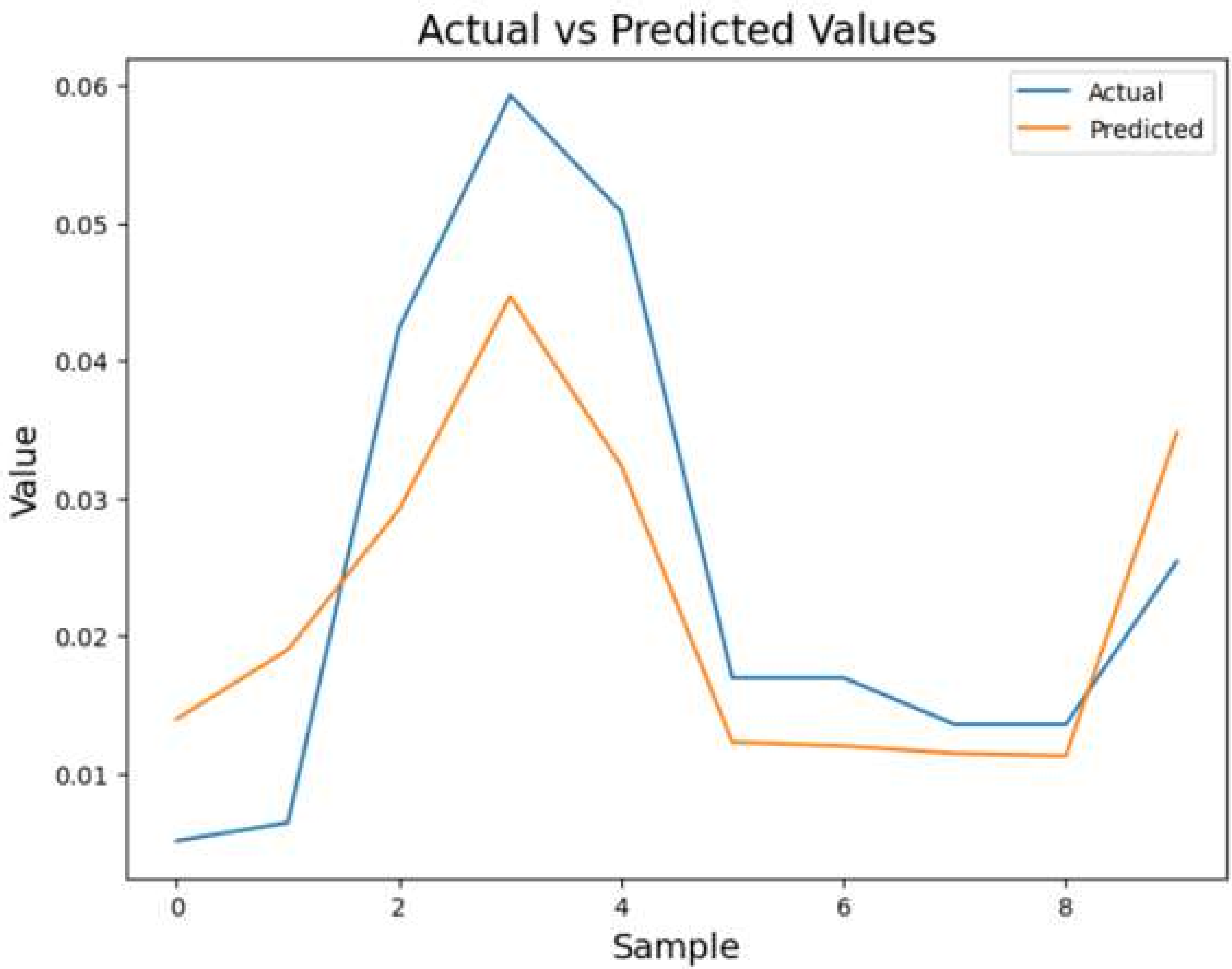


Figure 10: Line Graph

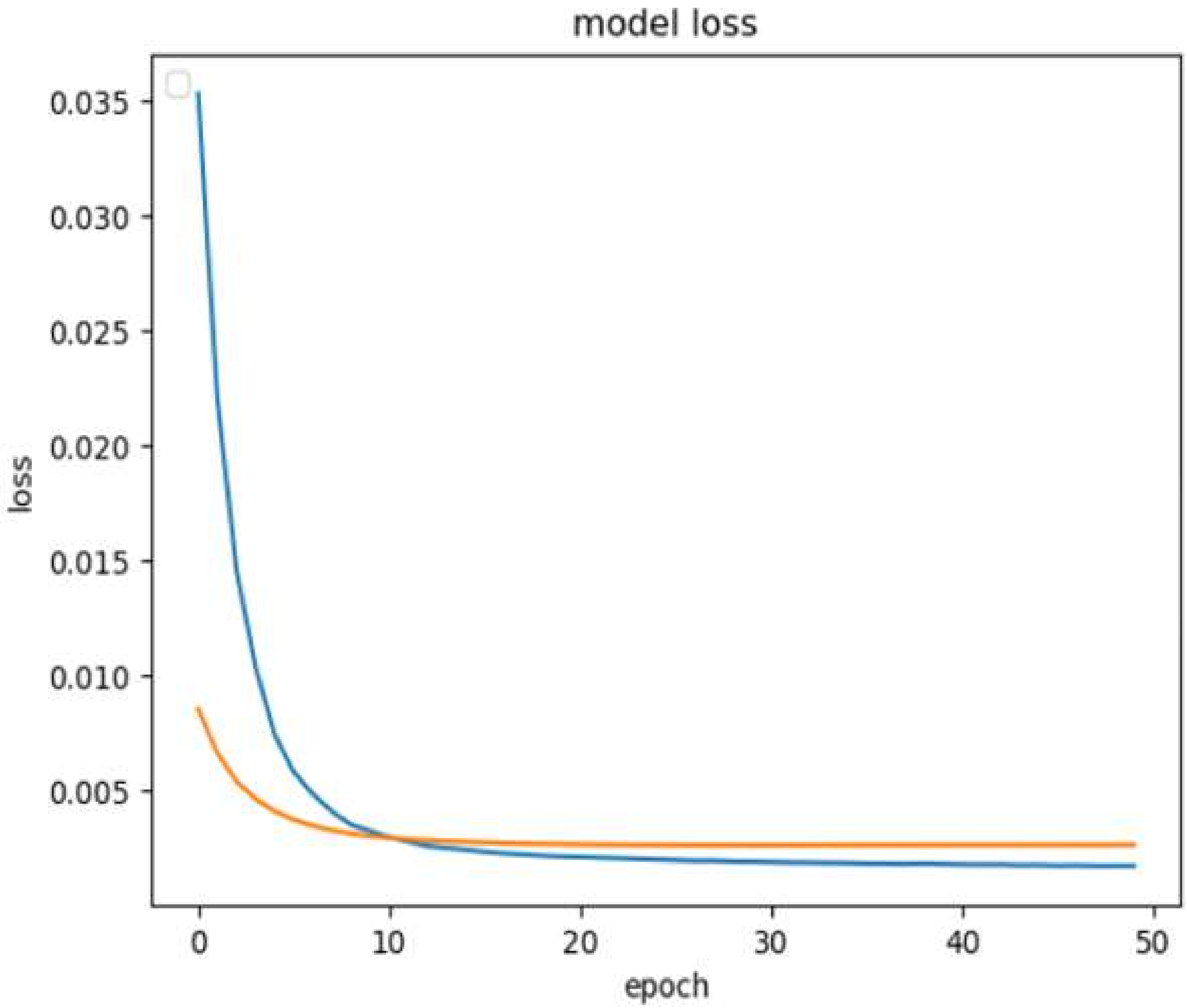
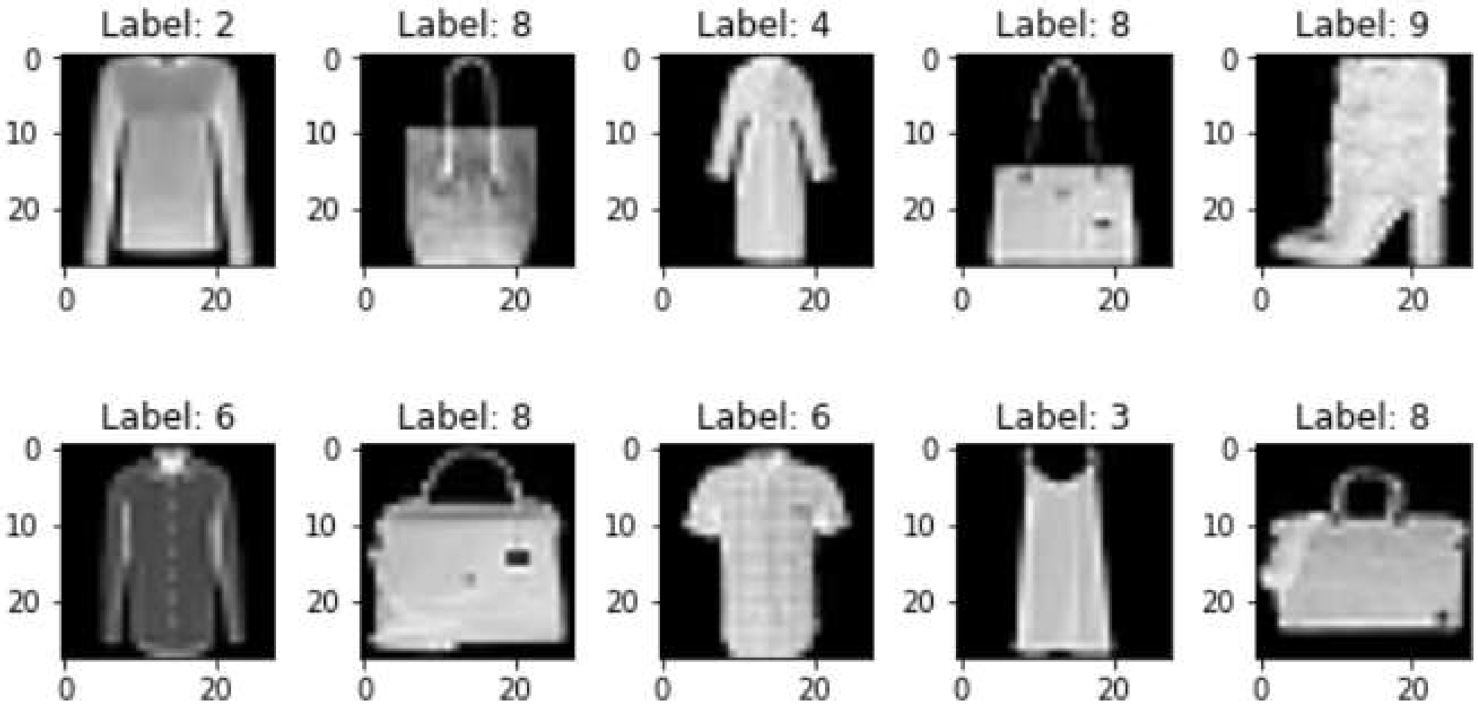


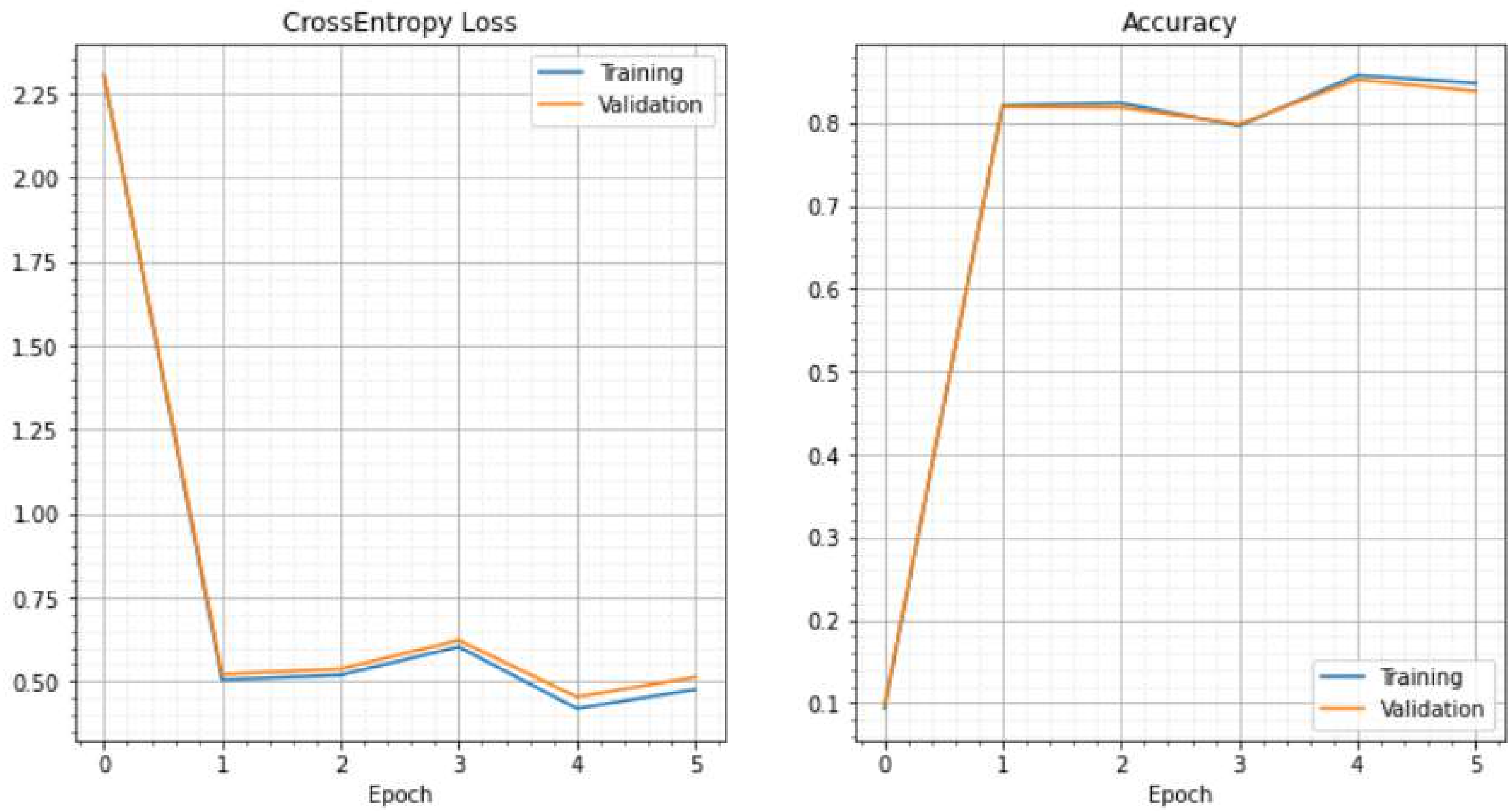
Figure 11: Line Graph of Model Accuracy

The line graph shows the loss which is the mean squared error (mse) on the y-axis and the number of epochs on the x-axis. It turns out that keeping the epoch at 50 instead of 10 and 20 improved the model`s performance even more and helped it to become more accurate.

Figure 6: Line Graph

The above figure is a line graph shows the general changes in price that takes place throughout the 3 years of the products listed in the dataset.


Label: 2
Label: 8
Label: 4
Label: 8
Label: 9
Label: 6
Label: 8
Label: 6
Label: 3
Label: 8



CrossEntropy Loss
Training
Validation
2.25
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0
1
2
3
4
5
Epoch
Accuracy
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
Training
Validation
0
1
2
3
4
5
Epoch