Appendix

Appendix 1: Sample CSV dataset used for training

	A	B	C	D	E	F
1	Date	Price	Open	High	Low	Change %
2	4-Jan-99	35.1205	35.087	35.5355	34.488	-0.0007
3	5-Jan-99	34.905	35.081	35.45	34.7	-0.0061
4	6-Jan-99	34.784	34.881	35.373	34.072	-0.0035
5	7-Jan-99	34.921	34.719	35.22	34.566	0.0039
6	8-Jan-99	35.0385	34.898	35.2095	34.744	0.0034
7	11-Jan-99	35.2415	35.004	35.3205	34.607	0.0058
8	12-Jan-99	35.453	35.21	35.53	34.904	0.006
9	13-Jan-99	35.829	35.41	36.206	35.252	0.0106
10	14-Jan-99	35.805	35.751	36.085	35.066	-0.0007
11	15-Jan-99	35.5975	35.728	36.1385	35.379	-0.0058
12	18-Jan-99	35.536	35.603	35.837	35.131	-0.0017
13	19-Jan-99	35.6985	35.514	35.8235	35.366	0.0046
14	20-Jan-99	35.972	35.669	36.161	35.262	0.0077
15	21-Jan-99	36.164	36.039	36.251	35.569	0.0053
16	22-Jan-99	36.2555	36.137	36.3715	35.945	0.0025
17	25-Jan-99	36.284	36.23	36.337	35.927	0.0008
18	26-Jan-99	36.623	36.279	36.732	36.127	0.0093
19	27-Jan-99	36.54	36.592	36.649	36.151	-0.0023
20	28-Jan-99	36.595	36.499	36.776	36.181	0.0015
21	29-Jan-99	36.782	36.506	37.021	36.346	0.0051
22	1-Feb-99	37.135	36.837	37.362	36.256	0.0096

Appendix 2: LSTM model sample code

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.metrics  import mean_squared_error
from keras.layers import LSTM
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt
from pandas import datetime
from keras.layers.core import Dense, Activation, Dropout
regressor = keras.Sequential()
regressor.add(LSTM(4, activation='relu', return_sequences=(True),input_shape=(1,11)))
regressor.add(LSTM(4,activation='sigmoid',return_sequences=(False)))
regressor.add(Dense(1) )
regressor.compile(optimizer='adam',loss="mean_squared_error")
dataset=pd.read_csv(r"C:\Users\Duni\Desktop\dollar.csv",parse_dates=['Date'], index_col='Date')

plt.title('Dataset')
dataset.plot()
plt.show()

print(dataset.isnull() )
dataset.dropna(inplace=True)
print(dataset)
#feature scaling (normalization)
from sklearn.preprocessing import MinMaxScaler
sc=MinMaxScaler()
traning_data=dataset.iloc[:,0:12].values
#getting the input and the outputs
x= traning_data[:, 1:12]
x=sc.fit_transform(x)

Xx = sc.inverse_transform(x)
plt.title('Traning Dataset')
plt.plot(Xx)
plt.ylabel('Price')
plt.xlabel('day')
plt.show()
y= traning_data[:, 0:1]
y=sc.fit_transform(y)
```

Appendix 3: Sample Code for Feature Selection

```
from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
from matplotlib import pyplot
import pandas as pd
from sklearn.datasets import make_regression
from sklearn.ensemble import RandomForestRegressor
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
dataframe = pd.read_csv(r"C:\Users\Duni\Desktop\pt\feature.csv", header=0)
X=dataframe [['Open','High','Low','Change %','MA_5','MA_10','CMA','EMA_0.1','EMA_0.3','12-Day EMA'
,'26-Day EMA','MACD','Signal Line','Upward Movement','Upward Movement','Downward Movement','Avg. 1
y=dataframe ['Price']  # Labels
y=y.astype('int')
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
sel = SelectFromModel(RandomForestClassifier(n_estimators = 100))
sel.fit(X_train, y_train)
sel.get_support()
selected_feat= X_train.columns[(sel.get_support())]
len(selected_feat)
```

Appendix 4: Sample Code for calculating technical indicators

```
df_26DayEMA=df_T['Price'].ewm(span=26).mean()
df_final=pd.read_csv(r"C:\Users\Duni\Desktop\euro.csv")
df_final['MA_5'] = df_forex1
df_final['MA_10'] = df_forex2
df_final['CMA'] = df_cumulative
df_final['EMA_0.1'] = df_EMA
df_final['EMA_0.3'] = df_EMA1
df_final['12-Day EMA'] = df_12DayEMA
df_final['26-Day EMA']=df_26DayEMA
df_MACD = df_final['12-Day EMA'] - df_final['26-Day EMA']
df_final['MACD']=df_MACD
df_SignalLine = df_final['MACD'].ewm(span=9).mean()
df_final['Signal Line'] = df_SignalLine
## calculating Relative Strength Index   momentum indicator
diff = df_final['Price'].diff()
up, down = diff.copy(), diff.copy()
up[up < 0] = 0
down[down > 0] = 0
df_final['Upward Movement'] = up
df_final['Downward Movement'] = abs(down)
df_14days_Up = df_final['Upward Movement'].rolling(14).mean() # using 14 day
df_14days_down = df_final['Downward Movement'].rolling(14).mean() # using 14
df_final['Avg. 14-Day Up Closes']=df_14days_Up
df_final['Avg. 14-Day Down Closes']=df_14days_down
df_RelativeStrength = df_final['Avg. 14-Day Up Closes'] / df_final['Avg. 14-D
df_final['Relative Strength'] = df_RelativeStrength
df_RSI = 100 - (100/(1+df_final['Relative Strength']   ))
```

Appendix 5: Sample 17 days Actual vs Predicted forex price for CZK/Euro Using LSTM

	Actual - NumPy object	final - NumPy object array
	0	0
0	26.061	26.3249
1	26.055	26.3935
2	26.009	26.4902
3	25.914	26.6055
4	25.87	26.4327
5	25.861	26.3323
6	25.827	26.3951
7	25.747	26.4947
8	25.653	26.505
9	25.715	26.0629
10	25.756	26.1462
11	25.725	26.3453
12	25.699	26.3192
13	25.656	26.3548
14	25.787	25.8778
15	25.818	26.1979
16	25.883	26.1105
17	25.875	26.3226