LAB Logbook

Github repository link:
https://github.com/Salimboevm/ML_Finance

Lab 1

For the lab work in week 1, students were asked to create a vector using np.arange method and do some changes on the vector to practice Numpy and Python.

My SID is 1919019, and because of that created 19 elements(last two digits). Then, transformed this vector into a 2 dimensional array with 1 row using the reshape() method. Then, used NumPy's empty_like() method and slicing to create an independent array and save the values of the vector to that independent array. And finally, checked the shape attribute values of both arrays and printed all results at the end of each step.

Results:

```
[302]: vector = np.arange(19)
       print(vector)

[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18]
```

```
[306]: vector = vector.reshape(19,1)
       print(vector)

[[ 0]
 [ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]
 [11]
 [12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]]
```

```
[308]: new_array_2d = np.empty_like(vector)
       new_array_2d[:,:] = vector
       print(new_array_2d)

[[ 0]
 [ 1]
 [ 2]
 [ 3]
 [ 4]
 [ 5]
 [ 6]
 [ 7]
 [ 8]
 [ 9]
 [10]
 [11]
 [12]
 [13]
 [14]
 [15]
 [16]
 [17]
 [18]]
```

```
[310]: print(vector.shape)
       print(new_array_2d.shape)

(19, 1)
(19, 1)
```

Lab 2

Lab 2 in Week 2, Pandas and its main functions were learnt. Requirements were using "adult_data_mini.csv" dataset and performing several operations.

Lab logbook requirement, n was determined as 9 (n=9) because of my student ID (last digit). Then data was grouped by "relationship" and "hours-per-week". Followed by "hours-per-week" column values were reduced by n=9. At this step, the function change_data(x) was created and used. To apply this function to the dataset, the apply() method was used and the original DataFrame was updated. Lastly, grouping by "relationship" and reduced "hours-per-week" was performed again.

Results:

Student ID: 1919019

[223]:
```
group_by_hours = data.groupby(['relationship', 'hours-per-week'])
group_by_hours.size()
```

[223]:
```
relationship   hours-per-week
Husband        13.0              1
               40.0              2
               45.0              1
               80.0              1
Not-in-family  16.0              1
               40.0              2
               50.0              2
Own-child      30.0              1
Wife           40.0              2
dtype: int64
```

[227]:
```
def change_data(x):
    return x - 9

data['hours-per-week'] = data['hours-per-week'].apply(change_data)
data
```

[227]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	Answer	IsHomeles
0	39	State-gov	77516.0	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	NaN	31.0	United-States	<=50K	Fals
1	50	Self-emp-not-inc	83311.0	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	4.0	United-States	<=50K	Fals
2	38	Private	215646.0	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	NaN	31.0	United-States	<=50K	Fals
3	53	Private	234721.0	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	NaN	31.0	United-States	<=50K	Fals
4	28	Private	338409.0	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	NaN	31.0	Cuba	<=50K	Fals
5	37	Private	284582.0	Masters	14.0	Married-civ-spouse	Exec-managerial	Wife	White	Female	0.0	NaN	31.0	United-States	<=50K	Fals
6	49	Private	160187.0	9th	5.0	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0.0	0.0	7.0	Jamaica	<=50K	Fals
7	52	Self-emp-not-inc	209642.0	HS-grad	9.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	36.0	United-States	>50K	Fals
8	31	Private	45781.0	Masters	14.0	Never-married	Prof-specialty	Not-in-family	White	Female	14084.0	NaN	41.0	United-States	>50K	Fals

8	31	Private	45781.0	Masters	14.0	Never-married	Prof-specialty	Not-in-family	White	Female	14084.0	NaN	41.0	United-States	>50K	Fals
10	37	Private	280464.0	Some-college	10.0	Married-civ-spouse	Exec-managerial	Husband	Black	Male	0.0	NaN	71.0	United-States	>50K	Fals
12	23	Private	122272.0	Bachelors	13.0	Never-married	Adm-clerical	Own-child	White	Female	0.0	NaN	21.0	United-States	<=50K	Fals
13	32	Private	205019.0	Assoc-acdm	12.0	Never-married	Sales	Not-in-family	Black	Male	0.0	NaN	41.0	United-States	<=50K	Fals
14	40	Private	121772.0	Assoc-voc	11.0	Married-civ-spouse	Craft-repair	Husband	Asian-Pac-Islander	Male	0.0	NaN	31.0	?	>50K	Fals
15	25	Private	NaN	Some-college	NaN	NaN	NaN	NaN	White	Male	0.0	NaN	NaN	NaN	NaN	Tru

[229]:
```
group_by_reduced_hours = data.groupby(['relationship', 'hours-per-week'])
group_by_reduced_hours.size()
```

[229]:
```
relationship   hours-per-week
Husband        4.0              1
               31.0             2
               36.0             1
               71.0             1
Not-in-family  7.0              1
               31.0             2
               41.0             2
Own-child      21.0             1
Wife           31.0             2
dtype: int64
```

[ ]:

Lab 3

Lab 3 in Week 3, a bicolour features interaction diagram drawing was requested as a requirement. Because of my student ID, the selected columns are the 1st and 9th columns (based on last two digits: 19).

According to the diagram, the visualisation shows the interaction between these two features using a bicolour scheme. The Seaborn pairplot() function with a hue parameter was used to create the bicolour effect, allowing for clear visual distinction between different categories in the dataset.

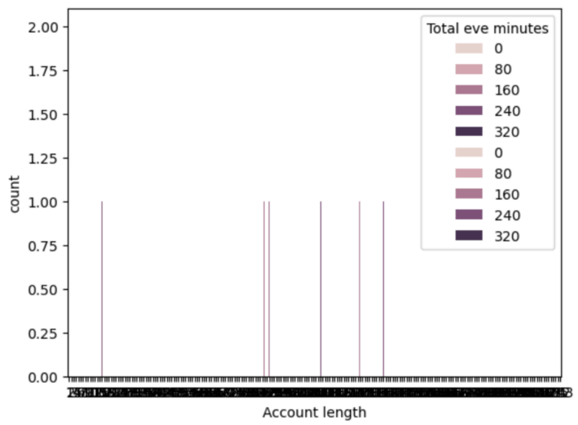
Results:

[ ]:

Student ID: 1919019

Type Markdown and LaTeX: α^2

[259]:
```
sns.countplot(x='Account length', hue='Total eve minutes', data=data);
plt.show()
```



Type Markdown and LaTeX: α^2

Lab 4

Lab 4 in Week 4, a Multilayer Perceptron (MLP) Neural Network was developed in the practical session. This was the first price forecast model, and the S&P-500 Index Prices dataset was used. For Lab 4, students were requested to create their own MLP model with two hidden layers.

Because of my student ID, the cells inside my hidden layers are 19 for the first hidden layer and 10 for the second hidden layer (approximately half of 19). There is also an output layer with one cell. After model creation, compiled the model and trained it with the same datasets from the practical session for 10 epochs. Finally, received the Mean Absolute Error (MAE) result.

Results:

```
[112]: model_2 = keras.Sequential([
           keras.layers.Dense(19, input_dim=500, activation=tf.nn.relu, kernel_initializer="normal"),

           keras.layers.Dense(10, activation='relu', kernel_initializer='normal'),

           #Output layer
           keras.layers.Dense(1)
       ])

       print(model_2.summary())
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
dense_8 (Dense)	(None, 19)	9,519
dense_9 (Dense)	(None, 10)	200
dense_10 (Dense)	(None, 1)	11

```
Total params: 9,730 (38.01 KB)
Trainable params: 9,730 (38.01 KB)
Non-trainable params: 0 (0.00 B)
None
```

```
[114]: # because my SID is 1919019 and last three digits are 019, I can't use 019 instead I used 19

[116]: model_2.compile(optimizer="adam", loss="mse", metrics=["mae"])

[118]: history_2 = model_2.fit(X_train, y_train, batch_size=10, epochs=10, validation_split=0.2, verbose=1)

Epoch 1/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 2s 729us/step - loss: 6.2650e-04 - mae: 0.0142 - val_loss: 0.0049 - val_mae: 0.0639
Epoch 2/10
```

Model Summary:

- Layer 1 (Dense): Output Shape (None, 19), Parameters: 9,519
- Layer 2 (Dense): Output Shape (None, 10), Parameters: 200
- Layer 3 (Dense): Output Shape (None, 1), Parameters: 11
- Total parameters: 9,730

```
[118]: history_2 = model_2.fit(X_train, y_train, batch_size=10, epochs=10, validation_split=0.2, verbose=1)

Epoch 1/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 2s 729us/step - loss: 6.2650e-04 - mae: 0.0142 - val_loss: 0.0049 - val_mae: 0.0639
Epoch 2/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 484us/step - loss: 1.0494e-04 - mae: 0.0079 - val_loss: 0.0015 - val_mae: 0.0332
Epoch 3/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 380us/step - loss: 7.9832e-05 - mae: 0.0069 - val_loss: 0.0013 - val_mae: 0.0304
Epoch 4/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 486us/step - loss: 6.8821e-05 - mae: 0.0064 - val_loss: 0.0012 - val_mae: 0.0287
Epoch 5/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 432us/step - loss: 6.1618e-05 - mae: 0.0059 - val_loss: 0.0018 - val_mae: 0.0375
Epoch 6/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 2s 573us/step - loss: 5.2630e-05 - mae: 0.0055 - val_loss: 8.2489e-04 - val_mae: 0.0234
Epoch 7/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 506us/step - loss: 4.9318e-05 - mae: 0.0053 - val_loss: 6.4474e-04 - val_mae: 0.0203
Epoch 8/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 553us/step - loss: 5.0440e-05 - mae: 0.0054 - val_loss: 6.4466e-04 - val_mae: 0.0205
Epoch 9/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 405us/step - loss: 4.8714e-05 - mae: 0.0052 - val_loss: 5.1350e-04 - val_mae: 0.0183
Epoch 10/10
2640/2640 ━━━━━━━━━━━━━━━━━━━━ 1s 417us/step - loss: 4.6789e-05 - mae: 0.0051 - val_loss: 3.1803e-04 - val_mae: 0.0149

[120]: mse_2, mae_2 = model_2.evaluate(X_test, y_test, verbose=0)

       print("Mean absolute error: %.5f" % mae_2)

Mean absolute error: 0.01643

[ ]:
```

Lab 5

Lab 5 in Week 5, a CNN code example was written in the practical session for price forecasting of Forex EUR/USD. For the lab logbook requirement, another CNN model creation was requested.

In this assignment, the convolutional core size (kernel_size) should be taken as 5, and batch_size should be 50. Additionally, the number of epochs should be determined according to Student IDs. Because of my student number, with Z=1 and Y=9, and after calculation it was 10. After compiling and training my model, I received the MAE value as a result.

Mean Absolute Error shows us the average of absolute differences between actual values and predicted values. The comparison between my CNN and the practical session CNN reveals how parameter differences (kernel_size, batch_size, and epochs) affect model performance. The CNN with lower MAE demonstrates better predictive capability.

Results:

Model: "sequential_2"

Layer (type)	Output Shape	Param #
conv1d_4 (Conv1D)	(None, 50, 50)	1,300
max_pooling1d_2 (MaxPooling1D)	(None, 7, 50)	0
conv1d_5 (Conv1D)	(None, 7, 100)	25,100
global_max_pooling1d_2 (GlobalMaxPooling1D)	(None, 100)	0
dense_4 (Dense)	(None, 25)	2,525
dense_5 (Dense)	(None, 2)	52

Total params: 28,977 (113.19 KB)
Trainable params: 28,977 (113.19 KB)
Non-trainable params: 0 (0.00 B)
None

[106]:
```
model_2.compile(optimizer="adam", loss="mse", metrics=["mae"])
```

[108]:
```
history_2 = model_2.fit(X_train, y_train, batch_size=50, epochs=10, validation_split=0.2, verbose=1)
```

```
Epoch 1/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 0.0025 - mae: 0.0277 - val_loss: 9.1150e-04 - val_mae: 0.0197
Epoch 2/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 7.5722e-04 - mae: 0.0188 - val_loss: 9.0916e-04 - val_mae: 0.0202
Epoch 3/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 7.1741e-04 - mae: 0.0181 - val_loss: 9.6782e-04 - val_mae: 0.0214
Epoch 4/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - loss: 7.1231e-04 - mae: 0.0180 - val_loss: 8.5019e-04 - val_mae: 0.0191
Epoch 5/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 7.0645e-04 - mae: 0.0179 - val_loss: 8.4837e-04 - val_mae: 0.0188
```

[100]:
```
#### Student ID: 1919019
```

[102]:
```
#Number of epochs: Z + Y = 10
#(Z = 1
#Y = 9)
```

[104]:
```
model_2 = keras.Sequential([
    keras.layers.Conv1D(50, 5, padding='same', input_shape=(50,5), activation=tf.nn.relu, kernel_initializer="normal"),

    keras.layers.MaxPooling1D(7),

    keras.layers.Conv1D(100, 5, padding='same', activation=tf.nn.relu, kernel_initializer="normal"),

    keras.layers.GlobalMaxPooling1D(),

    keras.layers.Dense(25, activation=tf.nn.relu, kernel_initializer="normal"),

    keras.layers.Dense(2)
])

print(model_2.summary())
```

Model: "sequential_2"

[108]:
```
history_2 = model_2.fit(X_train, y_train, batch_size=50, epochs=10, validation_split=0.2, verbose=1)
```

```
Epoch 1/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 0.0025 - mae: 0.0277 - val_loss: 9.1150e-04 - val_mae: 0.0197
Epoch 2/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 7.5722e-04 - mae: 0.0188 - val_loss: 9.0916e-04 - val_mae: 0.0202
Epoch 3/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - loss: 7.1741e-04 - mae: 0.0181 - val_loss: 9.6782e-04 - val_mae: 0.0214
Epoch 4/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 9s 2ms/step - loss: 7.1231e-04 - mae: 0.0180 - val_loss: 8.5019e-04 - val_mae: 0.0191
Epoch 5/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 9s 3ms/step - loss: 7.0645e-04 - mae: 0.0179 - val_loss: 8.4837e-04 - val_mae: 0.0188
Epoch 6/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 7.0028e-04 - mae: 0.0178 - val_loss: 9.6686e-04 - val_mae: 0.0213
Epoch 7/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 8s 2ms/step - loss: 6.9346e-04 - mae: 0.0176 - val_loss: 8.2854e-04 - val_mae: 0.0184
Epoch 8/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - loss: 6.9649e-04 - mae: 0.0177 - val_loss: 8.2802e-04 - val_mae: 0.0185
Epoch 9/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - loss: 6.9377e-04 - mae: 0.0177 - val_loss: 8.3583e-04 - val_mae: 0.0187
Epoch 10/10
3520/3520 ━━━━━━━━━━━━━━━━━━━━ 7s 2ms/step - loss: 6.9154e-04 - mae: 0.0176 - val_loss: 8.3504e-04 - val_mae: 0.0187
```

[109]:
```
mse, mae = model_2.evaluate(X_test, y_test, verbose=1)
print("Mean absolute error: %.5f" % mae)
```

```
936/936 ━━━━━━━━━━━━━━━━━━━━ 1s 601us/step - loss: 0.0013 - mae: 0.0245
Mean absolute error: 0.02453
```

[ ]:

Lab 6

Lab 6 in Week 6, detailed code for data preprocessing was written. For the Lab Logbook requirement, students were asked to plot the price chart of a part of the whole dataset showing 'High_Bid' and 'Low_Bid' prices using the iplot() library.

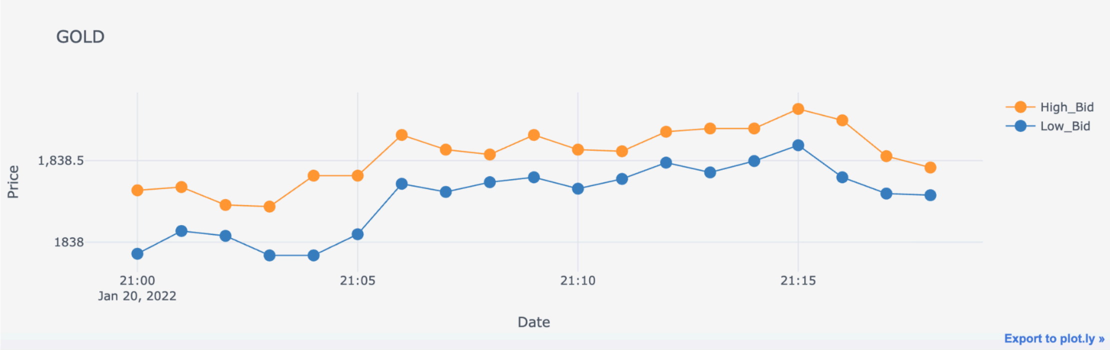
Because my student ID is 1919019, the start point = 19019 and the time period in minutes = 19 (last three digits = 019 = 19). In order for the time period to equal 19 minutes, the data in the graph will be from index 19019 to 19038 (19019 + 19 = 19038).

Results:

```
[285]: #Start point: 19019

       #Time period in minutes: 019

       # In order to the time period equals to 019 minutes, data will be from 19019 to 19038 (19019 + 019 = 19038)
```

```
[286]: data.iloc[19019:19038,:][['High_Bid', 'Low_Bid', 'Local_time_T', 'Volume_Ask','Volume_Bid']].iplot(
                                               x='Local_time_T', y=['High_Bid', 'Low_Bid'],
                                               mode='lines+markers',
                                               xTitle='Date', yTitle='Price',
                                               title='GOLD ')
```

```
/opt/anaconda3/lib/python3.12/site-packages/cufflinks/plotlytools.py:117: FutureWarning:

DatetimeIndex.format is deprecated and will be removed in a future version. Convert using index.astype(str) or index.map(formatter) instead.
```



Lab 7

Lab 7 in Week 7, LSTM and early stopping were explored. For the Lab Logbook requirement, a new LSTM model should be created with the following parameters:

- LSTM model parameter: ZY + 10 = 19 + 10 = 29
  (Because of my Student ID, ZY = 19)
- Epochs = 10
- Patience = 3

As a result, I obtained the MAE and MSE of my model and compared them with the model from the practical session.

Practical Session Results:

- Mean squared error (MSE): 0.000001760

- Mean absolute error (MAE): 0.001051122

My MSE & MAE:

- Mean squared error (MSE): 0.000006713
- Mean absolute error (MAE): 0.002116846

```
[154]: model = keras.Sequential([

           keras.layers.LSTM(29, activation='relu', input_shape=(50, 18)),

           keras.layers.Dense(2)

       ])
```

```
[156]: print(model.summary())
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
lstm_2 (LSTM)	(None, 29)	5,568
dense_2 (Dense)	(None, 2)	60

```
Total params: 5,628 (21.98 KB)
Trainable params: 5,628 (21.98 KB)
Non-trainable params: 0 (0.00 B)
None
```

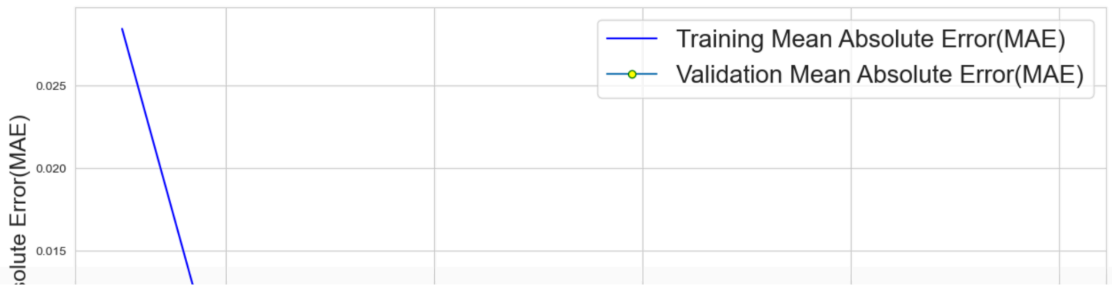
```
[158]: model.compile(optimizer="adam", loss="mse", metrics=["mae"])
```

```
[160]: es = EarlyStopping(monitor='val_loss', mode='min', patience=3, verbose=1)
       mc = ModelCheckpoint('best_model_LSTM_GOLD.keras', monitor='val_loss', mode='min', verbose=1, save_best_only=True)
```

```
[162]: history = model.fit(X_train, y_train, batch_size=20, epochs=10,
                           validation_split=0.1, shuffle=True,
                           verbose=1, callbacks=[es,mc])

       Epoch 1/10
       1205/1213 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - loss: 0.2037 - mae: 0.1252
```

```
[166]: print("Mean squared error (mse): %.9f " % (scores[0]))
       print("Mean absolute error (mae): %.9f " % (scores[1]))

       Mean squared error (mse): 0.000006713
       Mean absolute error (mae): 0.002116846
```

```
[167]: history_dict = history.history

       mae_values = history_dict['mae']
       val_mae_values = history_dict['val_mae']

       epochs = range(1, len(mae_values) + 1)
       plt.figure(num=1, figsize=(15,7))
       plt.plot(epochs, mae_values, 'b', label='Training Mean Absolute Error(MAE)')
       plt.plot(epochs, val_mae_values, marker='o', markeredgecolor='green', markerfacecolor='yellow', label='Validation Mean Absolute Error(MAE)')
       plt.xlabel('Epochs', size=18)
       plt.ylabel('Mean Absolute Error(MAE)', size=18)
       plt.legend()
       plt.show()
```



Lab 8

Lab 8 in Week 8, a mock test was conducted with Silver data. For Week 8, students were asked to create templates for the in-class test to be held in Week 9. I created three templates for CNN, LSTM, and MLP techniques.

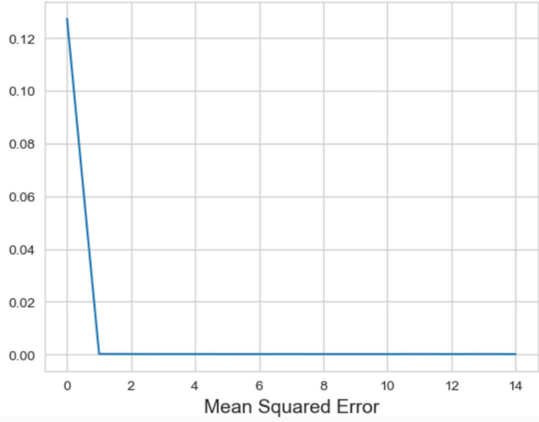
Results:

CNN

Plotting the Result Graphs

MSE Training Graphs

[107]:
```
#Training MSE (Mean Squared Error)

plt.plot(history.history['loss'])
plt.xlabel('Mean Squared Error', size=14)
plt.show()
```



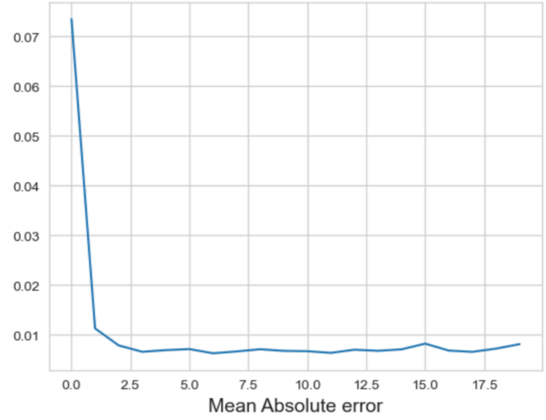

```
Epoch 2/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.0420e-04 - mae: 0.0110 - val_loss: 1.4942e-04 - val_mae: 0.0101
Epoch 3/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.3669e-04 - mae: 0.0118 - val_loss: 2.4954e-04 - val_mae: 0.0134
Epoch 4/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.0764e-04 - mae: 0.0110 - val_loss: 5.1185e-04 - val_mae: 0.0185
Epoch 5/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.9639e-04 - mae: 0.0135 - val_loss: 3.1542e-04 - val_mae: 0.0156
Epoch 6/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.4769e-04 - mae: 0.0123 - val_loss: 4.1447e-04 - val_mae: 0.0174
Epoch 7/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.4384e-04 - mae: 0.0147 - val_loss: 3.5681e-04 - val_mae: 0.0158
Epoch 8/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 4.1869e-04 - mae: 0.0162 - val_loss: 5.1637e-04 - val_mae: 0.0181
Epoch 9/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.1298e-04 - mae: 0.0140 - val_loss: 1.4145e-04 - val_mae: 0.0093
Epoch 10/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.5210e-04 - mae: 0.0124 - val_loss: 3.5750e-04 - val_mae: 0.0177
Epoch 11/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.1498e-04 - mae: 0.0113 - val_loss: 8.4602e-05 - val_mae: 0.0073
Epoch 12/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.8790e-04 - mae: 0.0129 - val_loss: 7.9444e-04 - val_mae: 0.0264
Epoch 13/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.1628e-04 - mae: 0.0140 - val_loss: 6.7907e-04 - val_mae: 0.0204
Epoch 14/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 2.8178e-04 - mae: 0.0132 - val_loss: 6.3462e-05 - val_mae: 0.0058
Epoch 15/15
143/143 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - loss: 3.6957e-04 - mae: 0.0152 - val_loss: 7.0861e-05 - val_mae: 0.0062
```

[378]:
```
mse, mae = model.evaluate(X_test, y_test, verbose=1)
print("Mean absolute error: %.5f" % mae)
print("Mean squared error: %.5f" % mse)
```

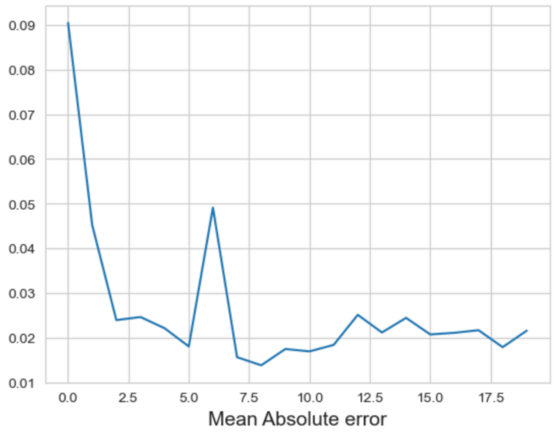
```
19/19 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step - loss: 1.9928e-04 - mae: 0.0115
Mean absolute error: 0.01154
Mean squared error: 0.00020
```

LSTM

[118]:
```python
#Training MAE (Mean Absolute Error)

plt.plot(history.history['mae'])
plt.xlabel('Mean Absolute error', size=14)
plt.show()
```



MLP



<u>Lab 9</u>

<u>Lab 10</u>

<u>Lab 11</u>

<u>Lab 12</u>