LAB Logbook

Lab 1

In [67]:
```
# import numpy module
import numpy as np
```

In [68]:
```
# set sid to string
sid = "2102636"
```

In [69]:
```
# calculate number of elements in vector
numberOfElements = int(sid[-2] + sid[-1])
# if number of elements is less than 10 add 100
if numberOfElements < 10:
  numberOfElements += 100
```

In [70]:
```
# create original vector
vector = np.arange(numberOfElements)
# print the vector
print(vector)
# print the vectors shape attribute value
print(vector.shape)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
 24 25 26 27 28 29 30 31 32 33 34 35]
(36,)
```

In [71]:
```
# reshape vector to matrix
matrix = np.reshape(vector, [1,numberOfElements])
# print matrix
print(matrix)
# print matrix shape attribute value
print(matrix.shape)
```

```
[[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
  24 25 26 27 28 29 30 31 32 33 34 35]]
(1, 36)
```

Lab 2

In [53]:
```
# hard code sid
sid = '2102636'
# take the last number of sid and convert to int
n = int(sid[-1])
# show our generated int n
n
```

Out[53]:
```
6
```

In [54]:
```
# group the data in the dataframe by relationship and hours per week
group_by_relationship = data.groupby(['relationship', 'hours-per-week'])
# display the size attribute of our group by dataframe
group_by_relationship.size()
```

Out[54]:
```
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
```

In [55]:
```
# reduce all values in hours per week by our number n
data['hours-per-week'] -= n
```

In [56]:
```
# # group the data in the dataframe by relationship and hours per week again
group_by_relationship = data.groupby(['relationship', 'hours-per-week'])
# display the size attribute of our group by dataframe to show hours per week has been reduced by n
group_by_relationship.size()
```

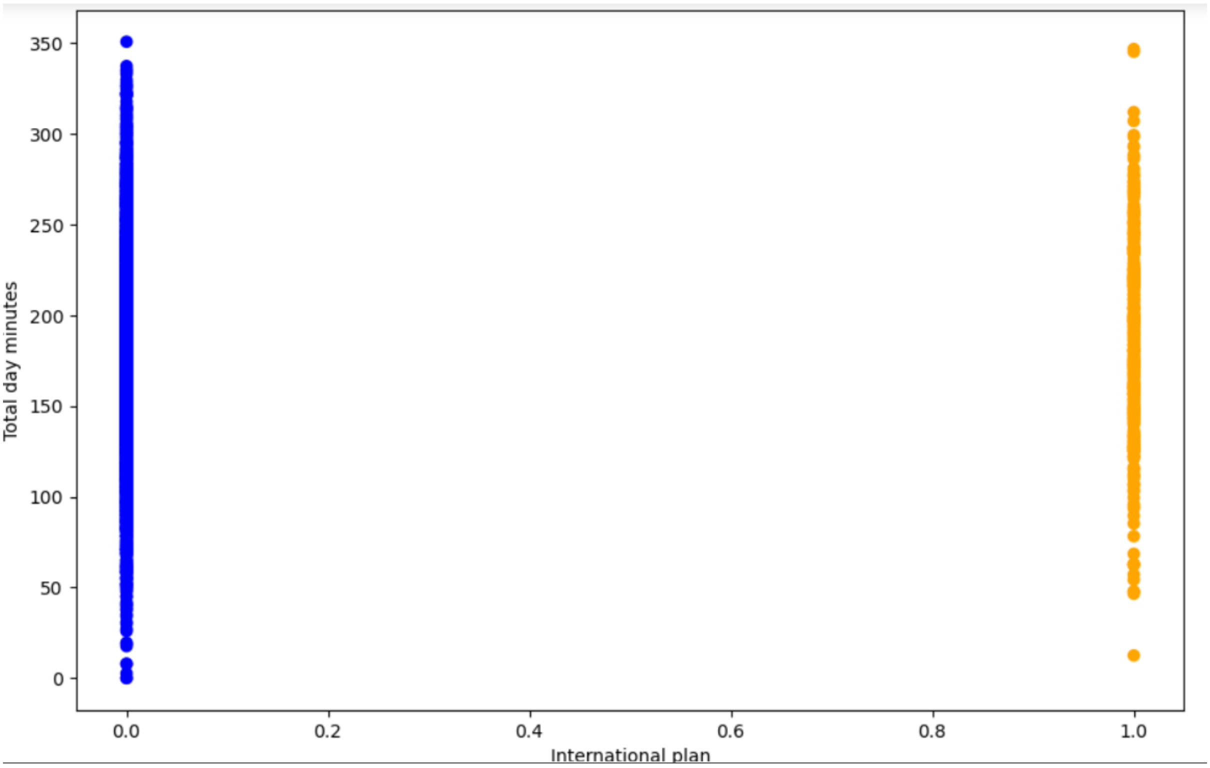
Out[56]:
```
relationship    hours-per-week
 Husband        7.0               1
                34.0              2
                39.0              1
                74.0              1
 Not-in-family  10.0              1
                34.0              2
                44.0              2
 Own-child      24.0              1
 Wife           34.0              2
dtype: int64
```

Lab 3

In [82]:
```python
# string used for sid
sid = '2102636'
# calculate the first and second column based on sid
first = int(sid[-2])
second = int(sid[-3])
```

In [91]:
```python
# make a map for the yes or no values to go to 0 or 1
d = {'No': 0, 'Yes': 1}
# rep;ace the values using the new map d
data = data.replace({'International plan': d})
# create a colour set based on the map
Clr = data['International plan'].map({0: 'blue', 1: 'orange'})
```

In [92]:
```python
# create a new figure
fig = plt.figure(figsize=(11, 7))
# plot the bicolour scatter plot with the colour set created
plt.scatter(data['International plan'], data['Total day minutes'], color=Clr);
# set x label as international plan
plt.xlabel('International plan');
# set y label as total day minutes
plt.ylabel('Total day minutes');
```



Lab 4

In [43]:
```
model = keras.Sequential([
    keras.layers.Dense(first_layer, input_dim=500, activation=tf.nn.relu, kernel_initializer="normal"),
    keras.layers.Dense(second_layer, activation='relu', kernel_initializer="normal"),
    keras.layers.Dense(1)
])

print(model.summary())
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
dense_5 (Dense)	(None, 636)	318,636
dense_6 (Dense)	(None, 318)	202,566
dense_7 (Dense)	(None, 1)	319

Total params: 521,521 (1.99 MB)

Trainable params: 521,521 (1.99 MB)

Non-trainable params: 0 (0.00 B)

None

:
```
mse, mae = model.evaluate(x_test, y_test, verbose=0)
print("Mean absolute error: %.5f" % mae)
print("Mean squared error: %.5f" % mse)
```

```
Mean absolute error: 0.06654
Mean squared error: 0.00494
```

Lab 5

In [42]:
```
model = 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.summary())
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv1d_2 (Conv1D)	(None, 50, 50)	1,300
max_pooling1d_1 (MaxPooling1D)	(None, 7, 50)	0
conv1d_3 (Conv1D)	(None, 7, 100)	25,100
global_max_pooling1d_1 (GlobalMaxPooling1D)	(None, 100)	0
dense_2 (Dense)	(None, 25)	2,525
dense_3 (Dense)	(None, 2)	52

Total params: 28,977 (113.19 KB)

Trainable params: 28,977 (113.19 KB)

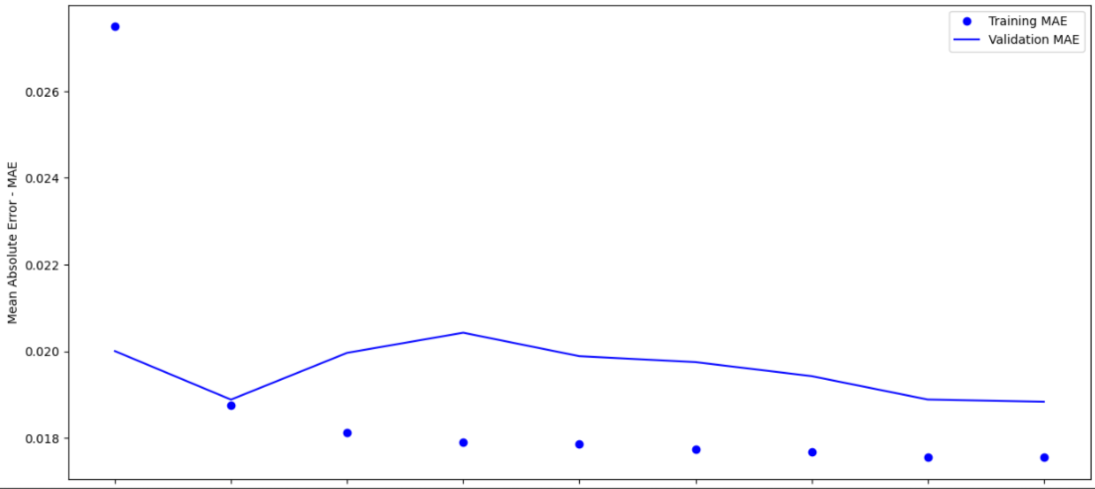
Non-trainable params: 0 (0.00 B)

In [63]:
```
mae = model.evaluate(X_test, y_test, verbose=1)[1]
print("Mean Absolute Error: %.5f" % mae)
```

```
936/936 ━━━━━━━━━━━━━━━━━━━━ 3s 3ms/step - loss: 0.0012 - mae: 0.0235
Mean Absolute Error: 0.02499
```

In [46]:
```
history_dict = history.history
mean_absolute_error_values = history_dict['mae']
val_mean_absolute_error_values = history_dict['val_mae']

epochs = range(1, len(mean_absolute_error_values) + 1)
plt.figure(num=1, figsize=(15,7))
plt.plot(epochs, mean_absolute_error_values, 'bo', label='Training MAE')
plt.plot(epochs, val_mean_absolute_error_values, 'b', label='Validation MAE')
plt.xlabel('Epochs')
plt.ylabel('Mean Absolute Error - MAE')
plt.legend()
plt.show()
```



Lab 6

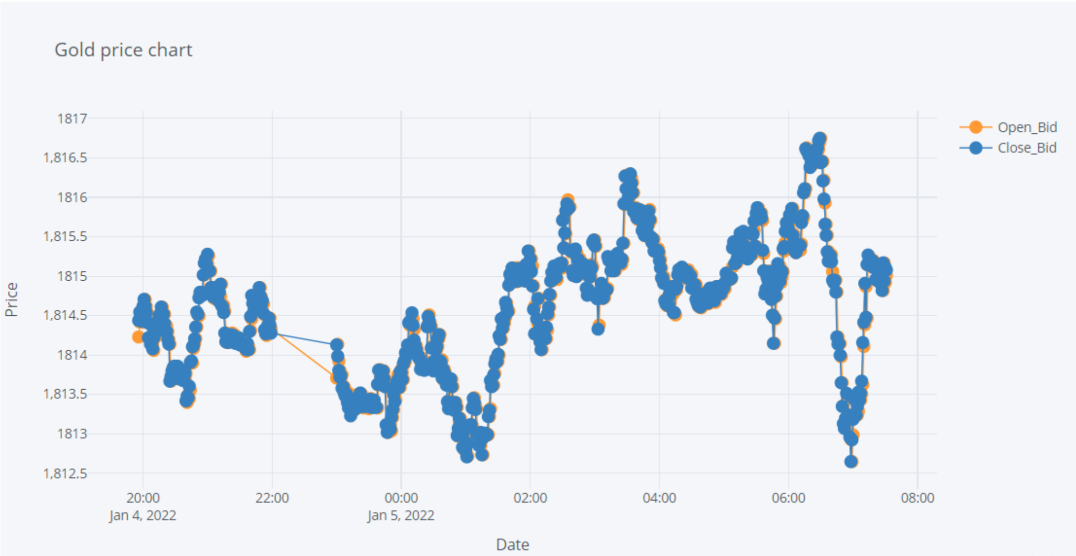
In [93]:
```
sid = '2102636'
start = int(sid[-5:])
time_period = int(sid[-3:])
```

In [94]:
```
my_data = data.iloc[start:start + time_period]
my_data.head()
```

Out[94]:

	Open_Bid	High_Bid	Low_Bid	Close_Bid	Volume_Bid	Volume_Ask	Volume_Delta	Volume_Delta_abs	Open_Delta	High_Delta	Low_Delta	Close_Delta
2636	1814.228	1814.485	1814.218	1814.438	0.01805	0.03280	0.01475	0.01475	0.374	0.330	0.354	0.344
2637	1814.418	1814.565	1814.315	1814.545	0.01123	0.02667	0.01544	0.01544	0.357	0.317	0.340	0.297
2638	1814.515	1814.698	1814.465	1814.518	0.01711	0.03936	0.02225	0.02225	0.327	0.307	0.330	0.334
2639	1814.518	1814.555	1814.268	1814.428	0.02543	0.06692	0.04149	0.04149	0.337	0.347	0.354	0.334
2640	1814.415	1814.665	1814.308	1814.618	0.01537	0.03485	0.01948	0.01948	0.337	0.310	0.344	0.317

In [95]:
```
my_data[['Open_Bid', 'Close_Bid', 'Local_time_T', 'Volume_Ask', 'Volume_Bid']].iplot(
    mode='lines+markers',
    x = 'Local_time_T',
    y=['Open_Bid', 'Close_Bid'],
    xTitle='Date',
    yTitle='Price',
    title='Gold price chart'
)
```



Lab 7

In [72]:
```
model = keras.Sequential([
    keras.layers.LSTM(LSTM_param, activation='relu', input_shape=(50, 18)),
    keras.layers.Dense(2)
])
print(model.summary())
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 46)	11,960
dense_1 (Dense)	(None, 2)	94

Total params: 12,054 (47.09 KB)

Trainable params: 12,054 (47.09 KB)

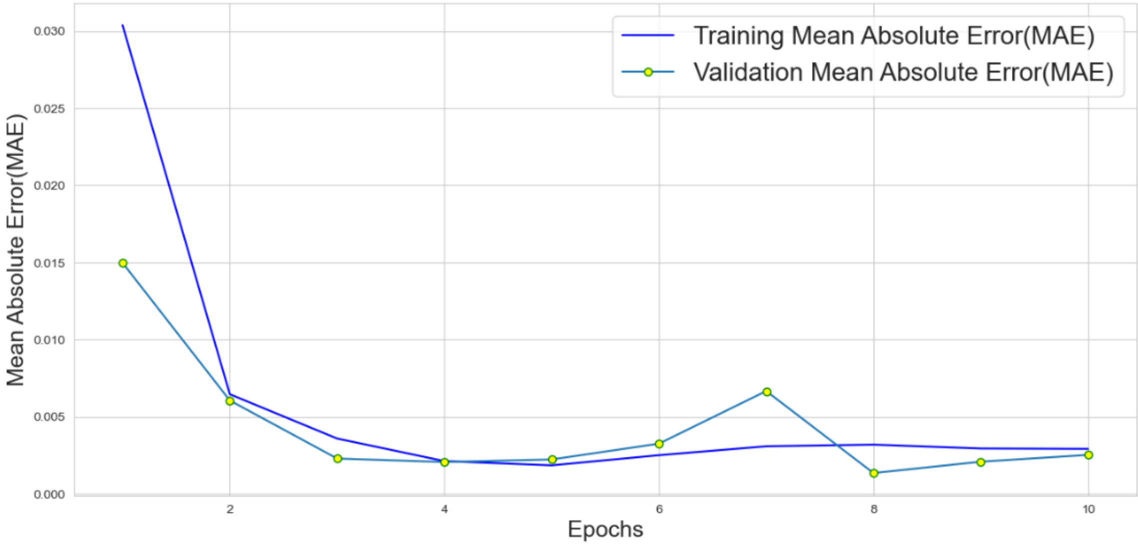
Non-trainable params: 0 (0.00 B)

None

In [77]:
```
print("Mean squared error (mse): %.9f " % (scores[0]))
print("Mean absolute error (mae): %.9f " % (scores[1]))
```

```
Mean squared error (mse): 0.000011572
Mean absolute error (mae): 0.002757165
```

```
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 E
plt.xlabel('Epochs', size=18)
plt.ylabel('Mean Absolute Error(MAE)', size=18)
plt.legend()
plt.show()
```



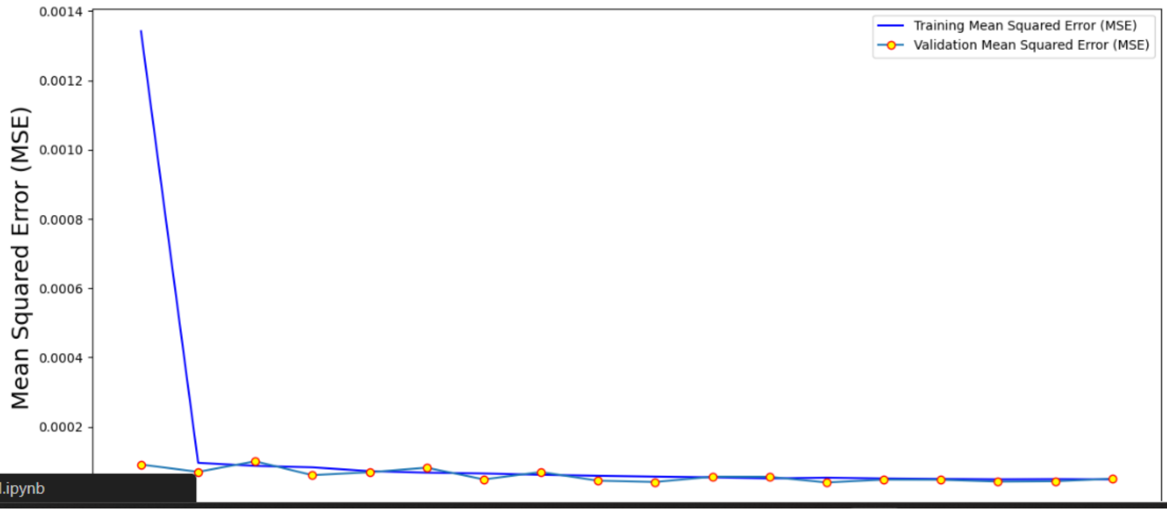
Lab 8

```
In [52]: print("Mean squared error (mse): %.9f " % (scores[0]))
         print("Mean absolute error (mae): %.9f " % (scores[1]))
```

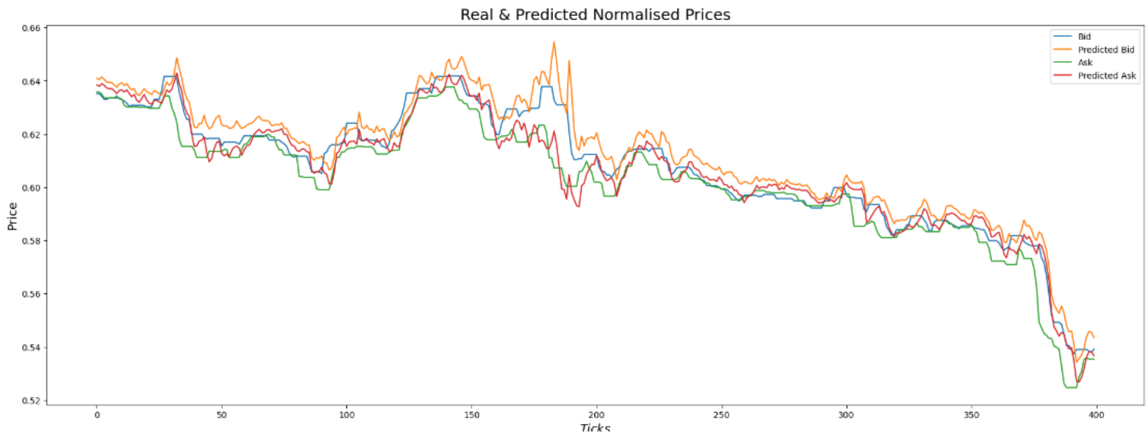
```
Mean squared error (mse): 0.000030586
Mean absolute error (mae): 0.003810658
```

```
mse_values = history_dict['loss']
val_mse_values = history_dict['val_loss']

epochs = range(1, len(mse_values) + 1)
plt.figure(num=1, figsize=(15,7))
plt.plot(epochs, mse_values, 'b', label='Training Mean Squared Error (MSE)')
plt.plot(epochs, val_mse_values, marker='o', markeredgecolor='red', markerfacecolor='yellow', label='Validation Mean Squared Erro
plt.xlabel('Epochs', size=18)
plt.ylabel('Mean Squared Error (MSE)', size=18)
plt.legend()
plt.show()
```

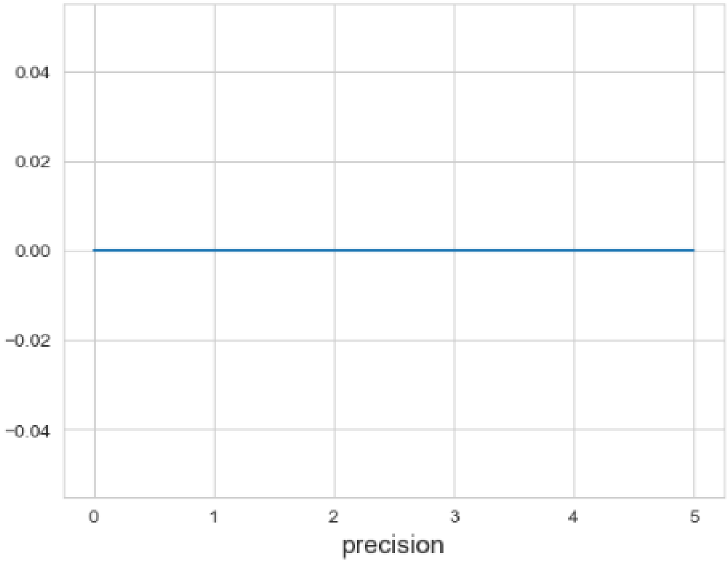


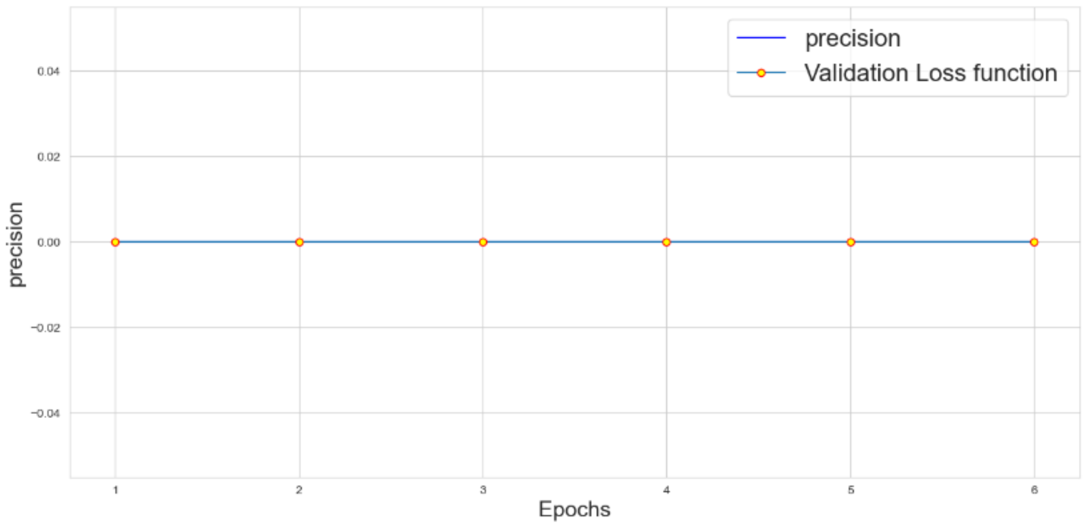
```
plt.title('Real & Predicted Normalised Prices', size=18)
plt.ylabel('Price', size=14)
plt.xlabel('$Ticks$', size=14)
plt.legend(loc='upper right')
plt.show()
```

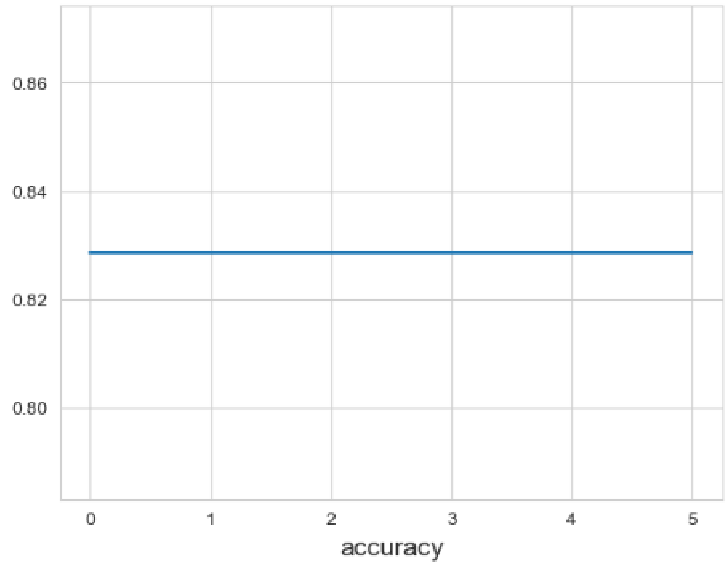


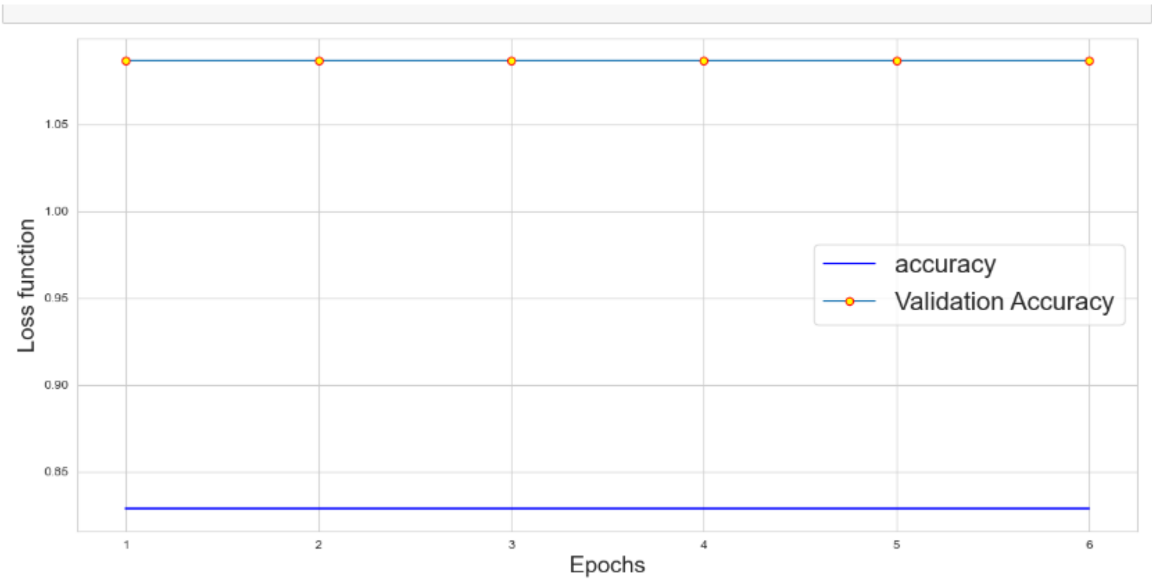
Lab 9

In [130]:
```
plt.plot(history.history['precision'])
plt.xlabel('precision', size=14)
plt.show()
```






0.86
0.84
0.82
0.80
0 1 2 3 4 5
accuracy

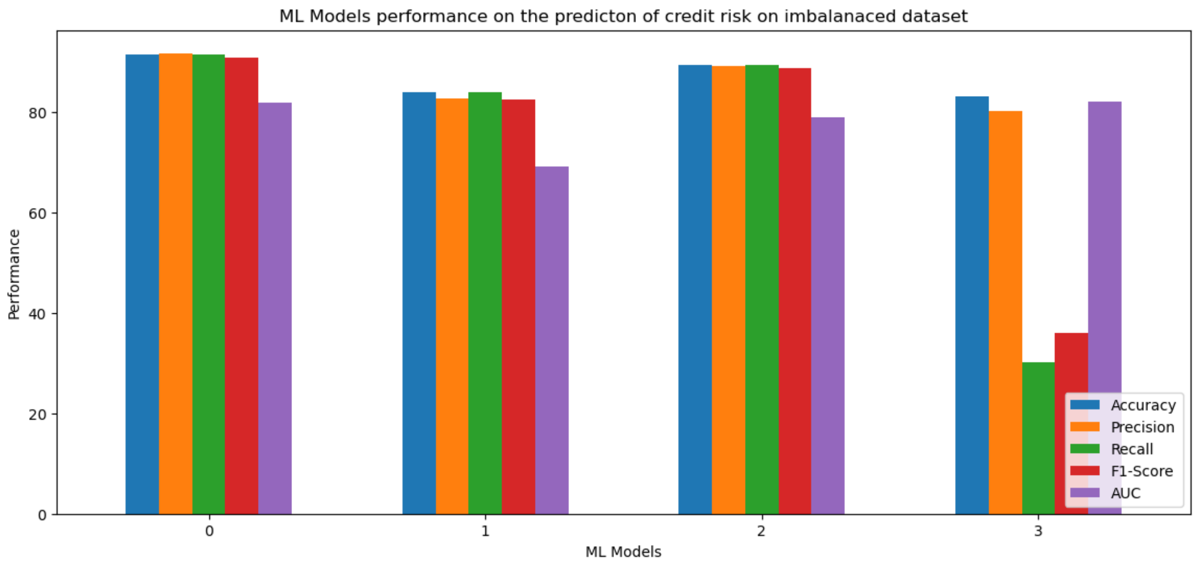

1.05
1.00
0.95
0.90
0.85
Loss function
1 2 3 4 5 6
Epochs
accuracy
Validation Accuracy

Lab 10

```
model.compile(optimizer='adam', metrics=['accuracy',
        metrics.Precision(name='precision'),
        metrics.Recall(name='recall'),
        tf.metrics.F1Score(average='macro'),
        metrics.AUC(name='auc')],
        loss='binary_crossentropy')
```

```
[281]: history = model.fit(X_train_imb, y_train, batch_size=32,
                    epochs=20,
                    validation_split=0.2,
                    shuffle=True,
                    verbose=1, callbacks=[es,mc])
```

```
Epoch 1/20
645/649 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.7975 - auc: 0.7085 - f1_score: 0.3554 - loss: 0.4936 - precision: 0.7058
- recall: 0.0890
Epoch 1: val_loss did not improve from 0.27766
649/649 ━━━━━━━━━━━━━━━━━━━━ 11s 11ms/step - accuracy: 0.7976 - auc: 0.7089 - f1_score: 0.3555 - loss: 0.4933 - precision: 0.70
64 - recall: 0.0898 - val_accuracy: 0.8330 - val_auc: 0.8172 - val_f1_score: 0.3633 - val_loss: 0.4205 - val_precision: 0.7062
- val_recall: 0.4240
Epoch 2/20
646/649 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8246 - auc: 0.8035 - f1_score: 0.3605 - loss: 0.4302 - precision: 0.7879
- recall: 0.2798
Epoch 2: val_loss did not improve from 0.27766
649/649 ━━━━━━━━━━━━━━━━━━━━ 7s 10ms/step - accuracy: 0.8246 - auc: 0.8036 - f1_score: 0.3605 - loss: 0.4301 - precision: 0.787
8 - recall: 0.2801 - val_accuracy: 0.8311 - val_auc: 0.8338 - val_f1_score: 0.3633 - val_loss: 0.4197 - val_precision: 0.8517 -
val_recall: 0.2893
Epoch 3/20
647/649 ━━━━━━━━━━━━━━━━━━━━ 0s 10ms/step - accuracy: 0.8332 - auc: 0.8143 - f1_score: 0.3601 - loss: 0.4123 - precision: 0.774
9 - recall: 0.3390
Epoch 3: val_loss did not improve from 0.27766
649/649 ━━━━━━━━━━━━━━━━━━━━ 7s 11ms/step - accuracy: 0.8331 - auc: 0.8142 - f1_score: 0.3601 - loss: 0.4124 - precision: 0.774
9 - recall: 0.3386 - val_accuracy: 0.8313 - val_auc: 0.8212 - val_f1_score: 0.3633 - val_loss: 0.3986 - val_precision: 0.7875 -
val_recall: 0.3284
Epoch 4/20
645/649 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8282 - auc: 0.8160 - f1_score: 0.3550 - loss: 0.4025 - precision: 0.7995
- recall: 0.2724
Epoch 4: val_loss did not improve from 0.27766
649/649 ━━━━━━━━━━━━━━━━━━━━ 7s 11ms/step - accuracy: 0.8282 - auc: 0.8160 - f1_score: 0.3550 - loss: 0.4025 - precision: 0.799
4 - recall: 0.2727 - val_accuracy: 0.8400 - val_auc: 0.8323 - val_f1_score: 0.3633 - val_loss: 0.3901 - val_precision: 0.8216 -
val_recall: 0.3562
Epoch 5/20
645/649 ━━━━━━━━━━━━━━━━━━━━ 0s 9ms/step - accuracy: 0.8375 - auc: 0.8341 - f1_score: 0.3605 - loss: 0.3932 - precision: 0.7939
- recall: 0.3546
Epoch 5: val_loss did not improve from 0.27766
```



GitHub

https://github.com/Freddie-Faulkner03/ML-In-Finance-2102636