Train a model with Iris data using XGBoost algorithm

Model is trained with XGBoost installed in notebook instance

In the later examples, we will train using SageMaker's XGBoost algorithm

In [1]:

```
# Install xgboost in notebook instance.
#### Command to install xgboost
!pip install xgboost
```

```
Looking in indexes: https://pypi.org/simple, https://pip.repos.neuron.amazonaws.com
Requirement already satisfied: xgboost in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (1.7.6)
Requirement already satisfied: numpy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgb
oost) (1.22.3)
Requirement already satisfied: scipy in /home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages (from xgb
oost) (1.10.1)
```

In [2]:

```
import sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import itertools
import xgboost as xgb

from sklearn import preprocessing
from sklearn.metrics import classification_report, confusion_matrix
```

In [3]:

```
column_list_file = 'iris_train_column_list.txt'
train_file = 'iris_train.csv'
validation_file = 'iris_validation.csv'
```

In [4]:

```
columns = ''
with open(column_list_file,'r') as f:
    columns = f.read().split(',')
```

In [5]:

```
columns
```

Out[5]:
```
['encoded_class', 'sepal_length', 'sepal_width', 'petal_length', 'petal_width']
```

In [6]:
```python
# Encode Class Labels to integers
# Labeled Classes
labels=[0,1,2]
classes = ['Iris-setosa', 'Iris-versicolor', 'Iris-virginica']
le = preprocessing.LabelEncoder()
le.fit(classes)
```

Out[6]:
```
▾ LabelEncoder
LabelEncoder()
```

In [7]:
```python
# Specify the column names as the file does not have column header
df_train = pd.read_csv(train_file,names=columns)
df_validation = pd.read_csv(validation_file,names=columns)
```

In [8]:
```python
df_train.head()
```

Out[8]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width
0	1	5.8	2.7	3.9	1.2
1	2	6.1	2.6	5.6	1.4
2	2	5.8	2.8	5.1	2.4
3	0	4.4	3.2	1.3	0.2
4	2	7.2	3.6	6.1	2.5

In [9]:
```python
df_validation.head()
```

Out[9]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width
0	1	5.8	2.7	4.1	1.0
1	0	4.8	3.4	1.6	0.2
2	1	6.0	2.2	4.0	1.0
3	2	6.4	3.1	5.5	1.8
4	2	6.7	2.5	5.8	1.8

In [10]:

```
X_train = df_train.iloc[:,1:] # Features: 1st column onwards
y_train = df_train.iloc[:,0].ravel() # Target: 0th column

X_validation = df_validation.iloc[:,1:]
y_validation = df_validation.iloc[:,0].ravel()
```

In [11]:

```
# Launch a classifier
# XGBoost Training Parameter Reference:
#   https://xgboost.readthedocs.io/en/latest/parameter.html

classifier = xgb.XGBClassifier(objective="multi:softmax",
                               num_class=3,
                               n_estimators=100)
```

In [12]:

```
classifier
```

Out[12]:

▾ XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [13]:

```
classifier.fit(X_train,
               y_train,
               eval_set = [(X_train, y_train), (X_validation, y_validation)],
               eval_metric=['mlogloss'])
```

```
[0]	validation_0-mlogloss:0.73876	validation_1-mlogloss:0.74994
[1]	validation_0-mlogloss:0.52787	validation_1-mlogloss:0.55401
[2]	validation_0-mlogloss:0.38959	validation_1-mlogloss:0.42612
[3]	validation_0-mlogloss:0.29429	validation_1-mlogloss:0.34328
[4]	validation_0-mlogloss:0.22736	validation_1-mlogloss:0.29000
[5]	validation_0-mlogloss:0.17920	validation_1-mlogloss:0.24961
[6]	validation_0-mlogloss:0.14403	validation_1-mlogloss:0.22234
[7]	validation_0-mlogloss:0.11664	validation_1-mlogloss:0.20338
[8]	validation_0-mlogloss:0.09668	validation_1-mlogloss:0.18999
[9]	validation_0-mlogloss:0.08128	validation_1-mlogloss:0.18190
[10]	validation_0-mlogloss:0.06783	validation_1-mlogloss:0.17996
[11]	validation_0-mlogloss:0.05794	validation_1-mlogloss:0.18029
[12]	validation_0-mlogloss:0.05011	validation_1-mlogloss:0.18306
[13]	validation_0-mlogloss:0.04428	validation_1-mlogloss:0.18471
[14]	validation_0-mlogloss:0.03993	validation_1-mlogloss:0.18693
[15]	validation_0-mlogloss:0.03615	validation_1-mlogloss:0.18553
[16]	validation_0-mlogloss:0.03310	validation_1-mlogloss:0.18571
[17]	validation_0-mlogloss:0.03065	validation_1-mlogloss:0.18615
[18]	validation_0-mlogloss:0.02874	validation_1-mlogloss:0.18930
[19]	validation_0-mlogloss:0.02739	validation_1-mlogloss:0.18989
[20]	validation_0-mlogloss:0.02639	validation_1-mlogloss:0.19251
[21]	validation_0-mlogloss:0.02583	validation_1-mlogloss:0.19567
[22]	validation_0-mlogloss:0.02513	validation_1-mlogloss:0.19760
[23]	validation_0-mlogloss:0.02444	validation_1-mlogloss:0.19690
[24]	validation_0-mlogloss:0.02398	validation_1-mlogloss:0.19946
[25]	validation_0-mlogloss:0.02340	validation_1-mlogloss:0.20132
[26]	validation_0-mlogloss:0.02287	validation_1-mlogloss:0.20281
[27]	validation_0-mlogloss:0.02250	validation_1-mlogloss:0.20464
[28]	validation_0-mlogloss:0.02217	validation_1-mlogloss:0.20638
[29]	validation_0-mlogloss:0.02185	validation_1-mlogloss:0.20661
[30]	validation_0-mlogloss:0.02150	validation_1-mlogloss:0.20768
[31]	validation_0-mlogloss:0.02122	validation_1-mlogloss:0.20791
[32]	validation_0-mlogloss:0.02091	validation_1-mlogloss:0.21019
[33]	validation_0-mlogloss:0.02064	validation_1-mlogloss:0.21058
[34]	validation_0-mlogloss:0.02038	validation_1-mlogloss:0.21031
[35]	validation_0-mlogloss:0.02010	validation_1-mlogloss:0.21248
[36]	validation_0-mlogloss:0.01989	validation_1-mlogloss:0.21323
[37]	validation_0-mlogloss:0.01964	validation_1-mlogloss:0.21301
[38]	validation_0-mlogloss:0.01945	validation_1-mlogloss:0.21486
[39]	validation_0-mlogloss:0.01927	validation_1-mlogloss:0.21497
[40]	validation_0-mlogloss:0.01907	validation_1-mlogloss:0.21486
[41]	validation_0-mlogloss:0.01890	validation_1-mlogloss:0.21675
```

```
[42]	validation_0-mlogloss:0.01872	validation_1-mlogloss:0.21662
[43]	validation_0-mlogloss:0.01856	validation_1-mlogloss:0.21786
[44]	validation_0-mlogloss:0.01839	validation_1-mlogloss:0.21843
[45]	validation_0-mlogloss:0.01824	validation_1-mlogloss:0.21844
[46]	validation_0-mlogloss:0.01809	validation_1-mlogloss:0.21965
[47]	validation_0-mlogloss:0.01794	validation_1-mlogloss:0.21966
[48]	validation_0-mlogloss:0.01780	validation_1-mlogloss:0.22028
[49]	validation_0-mlogloss:0.01766	validation_1-mlogloss:0.22134
[50]	validation_0-mlogloss:0.01752	validation_1-mlogloss:0.22137
[51]	validation_0-mlogloss:0.01740	validation_1-mlogloss:0.22236
[52]	validation_0-mlogloss:0.01727	validation_1-mlogloss:0.22295
[53]	validation_0-mlogloss:0.01715	validation_1-mlogloss:0.22300
[54]	validation_0-mlogloss:0.01703	validation_1-mlogloss:0.22396
[55]	validation_0-mlogloss:0.01692	validation_1-mlogloss:0.22390
[56]	validation_0-mlogloss:0.01681	validation_1-mlogloss:0.22542
[57]	validation_0-mlogloss:0.01670	validation_1-mlogloss:0.22549
[58]	validation_0-mlogloss:0.01659	validation_1-mlogloss:0.22555
[59]	validation_0-mlogloss:0.01650	validation_1-mlogloss:0.22563
[60]	validation_0-mlogloss:0.01641	validation_1-mlogloss:0.22627
[61]	validation_0-mlogloss:0.01632	validation_1-mlogloss:0.22637
[62]	validation_0-mlogloss:0.01623	validation_1-mlogloss:0.22647
[63]	validation_0-mlogloss:0.01620	validation_1-mlogloss:0.22713
[64]	validation_0-mlogloss:0.01617	validation_1-mlogloss:0.22724
[65]	validation_0-mlogloss:0.01614	validation_1-mlogloss:0.22734
[66]	validation_0-mlogloss:0.01611	validation_1-mlogloss:0.22797
[67]	validation_0-mlogloss:0.01609	validation_1-mlogloss:0.22796
[68]	validation_0-mlogloss:0.01606	validation_1-mlogloss:0.22856
[69]	validation_0-mlogloss:0.01604	validation_1-mlogloss:0.22867
[70]	validation_0-mlogloss:0.01602	validation_1-mlogloss:0.22867
[71]	validation_0-mlogloss:0.01599	validation_1-mlogloss:0.22926
[72]	validation_0-mlogloss:0.01597	validation_1-mlogloss:0.22926
[73]	validation_0-mlogloss:0.01595	validation_1-mlogloss:0.22983
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/xgboost/sklearn.py:835: UserWarning: `eval_metric
` in `fit` method is deprecated for better compatibility with scikit-learn, use `eval_metric` in constructor or`set_
params` instead.
  warnings.warn(
[74]	validation_0-mlogloss:0.01592	validation_1-mlogloss:0.22994
[75]	validation_0-mlogloss:0.01590	validation_1-mlogloss:0.22994
[76]	validation_0-mlogloss:0.01588	validation_1-mlogloss:0.23050
[77]	validation_0-mlogloss:0.01586	validation_1-mlogloss:0.23051
[78]	validation_0-mlogloss:0.01584	validation_1-mlogloss:0.23106
[79]	validation_0-mlogloss:0.01582	validation_1-mlogloss:0.23116
[80]	validation_0-mlogloss:0.01580	validation_1-mlogloss:0.23117
[81]	validation_0-mlogloss:0.01578	validation_1-mlogloss:0.23171
[82]	validation_0-mlogloss:0.01576	validation_1-mlogloss:0.23172
[83]	validation_0-mlogloss:0.01574	validation_1-mlogloss:0.23224
[84]	validation_0-mlogloss:0.01573	validation_1-mlogloss:0.23225
[85]	validation_0-mlogloss:0.01571	validation_1-mlogloss:0.23277
[86]	validation_0-mlogloss:0.01569	validation_1-mlogloss:0.23287
[87]	validation_0-mlogloss:0.01567	validation_1-mlogloss:0.23288
[88]	validation_0-mlogloss:0.01566	validation_1-mlogloss:0.23339
[89]	validation_0-mlogloss:0.01564	validation_1-mlogloss:0.23340
[90]	validation_0-mlogloss:0.01562	validation_1-mlogloss:0.23390
[91]	validation_0-mlogloss:0.01561	validation_1-mlogloss:0.23390
[92]	validation_0-mlogloss:0.01559	validation_1-mlogloss:0.23439
[93]	validation_0-mlogloss:0.01558	validation_1-mlogloss:0.23449
[94]	validation_0-mlogloss:0.01556	validation_1-mlogloss:0.23450
[95]	validation_0-mlogloss:0.01555	validation_1-mlogloss:0.23498
[96]	validation_0-mlogloss:0.01553	validation_1-mlogloss:0.23499
[97]	validation_0-mlogloss:0.01552	validation_1-mlogloss:0.23546
[98]	validation_0-mlogloss:0.01550	validation_1-mlogloss:0.23556
[99]	validation_0-mlogloss:0.01549	validation_1-mlogloss:0.23556
```

Out[13]: ▾ XGBClassifier

```
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

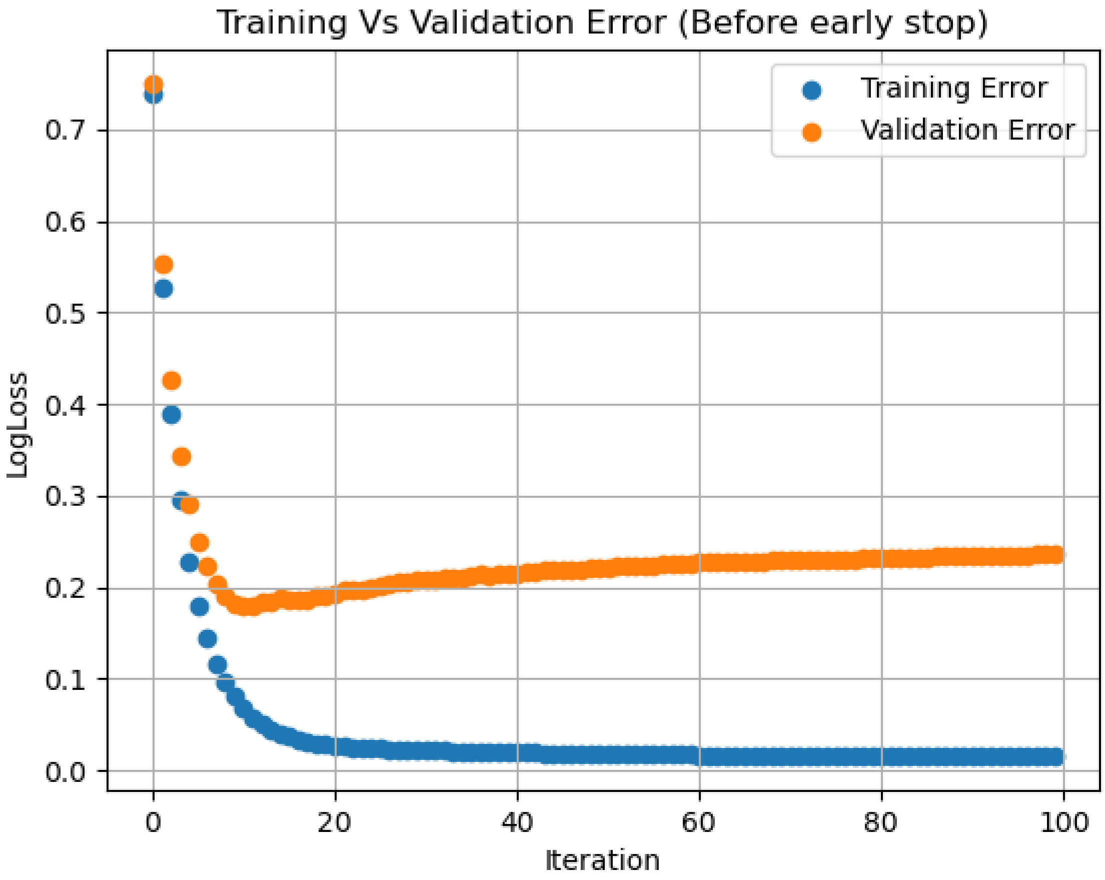
In [14]:

```
eval_result_before_early_stop = classifier.evals_result()
training_rounds = range(len(eval_result_before_early_stop['validation_0']['mlogloss']))
print(f"training_rounds: {training_rounds}")
```

```
training_rounds: range(0, 100)
```

In [15]:

```
plt.scatter(x=training_rounds,
            y=eval_result_before_early_stop[
                'validation_0']['mlogloss'],
            label='Training Error')
plt.scatter(x=training_rounds,
            y=eval_result_before_early_stop[
                'validation_1']['mlogloss'],label='Validation Error')
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('LogLoss')
plt.title('Training Vs Validation Error (Before early stop)')
plt.legend()
plt.show()
```



In [16]:

```
classifier.fit(X_train,
               y_train,
               eval_set = [(X_train, y_train), (X_validation, y_validation)],
               eval_metric=['mlogloss'],
               early_stopping_rounds=10)

# early_stopping_rounds - needs to be passed in as a hyperparameter in SageMaker XGBoost implementation
# "The model trains until the validation score stops improving.
# Validation error needs to decrease at least every early_stopping_rounds to continue training.
# Amazon SageMaker hosting uses the best model for inference."
```

```
[0]	validation_0-mlogloss:0.73876	validation_1-mlogloss:0.74994
[1]	validation_0-mlogloss:0.52787	validation_1-mlogloss:0.55401
[2]	validation_0-mlogloss:0.38959	validation_1-mlogloss:0.42612
[3]	validation_0-mlogloss:0.29429	validation_1-mlogloss:0.34328
[4]	validation_0-mlogloss:0.22736	validation_1-mlogloss:0.29000
[5]	validation_0-mlogloss:0.17920	validation_1-mlogloss:0.24961
[6]	validation_0-mlogloss:0.14403	validation_1-mlogloss:0.22234
[7]	validation_0-mlogloss:0.11664	validation_1-mlogloss:0.20338
[8]	validation_0-mlogloss:0.09668	validation_1-mlogloss:0.18999
[9]	validation_0-mlogloss:0.08128	validation_1-mlogloss:0.18190
[10]	validation_0-mlogloss:0.06783	validation_1-mlogloss:0.17996
[11]	validation_0-mlogloss:0.05794	validation_1-mlogloss:0.18029
[12]	validation_0-mlogloss:0.05011	validation_1-mlogloss:0.18306
[13]	validation_0-mlogloss:0.04428	validation_1-mlogloss:0.18471
[14]	validation_0-mlogloss:0.03993	validation_1-mlogloss:0.18693
[15]	validation_0-mlogloss:0.03615	validation_1-mlogloss:0.18553
[16]	validation_0-mlogloss:0.03310	validation_1-mlogloss:0.18571
[17]	validation_0-mlogloss:0.03065	validation_1-mlogloss:0.18615
[18]	validation_0-mlogloss:0.02874	validation_1-mlogloss:0.18930
[19]	validation_0-mlogloss:0.02739	validation_1-mlogloss:0.18989
```

```
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/xgboost/sklearn.py:835: UserWarning: `eval_metric
` in `fit` method is deprecated for better compatibility with scikit-learn, use `eval_metric` in constructor or`set_
params` instead.
  warnings.warn(
/home/ec2-user/anaconda3/envs/python3/lib/python3.10/site-packages/xgboost/sklearn.py:835: UserWarning: `early_stopp
ing_rounds` in `fit` method is deprecated for better compatibility with scikit-learn, use `early_stopping_rounds` in
constructor or`set_params` instead.
  warnings.warn(
```

Out[16]:

```
▾                              XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
```

In [17]:

```
eval_result = classifier.evals_result()
```

In [18]:

```
training_rounds = range(len(eval_result['validation_0']['mlogloss']))
```

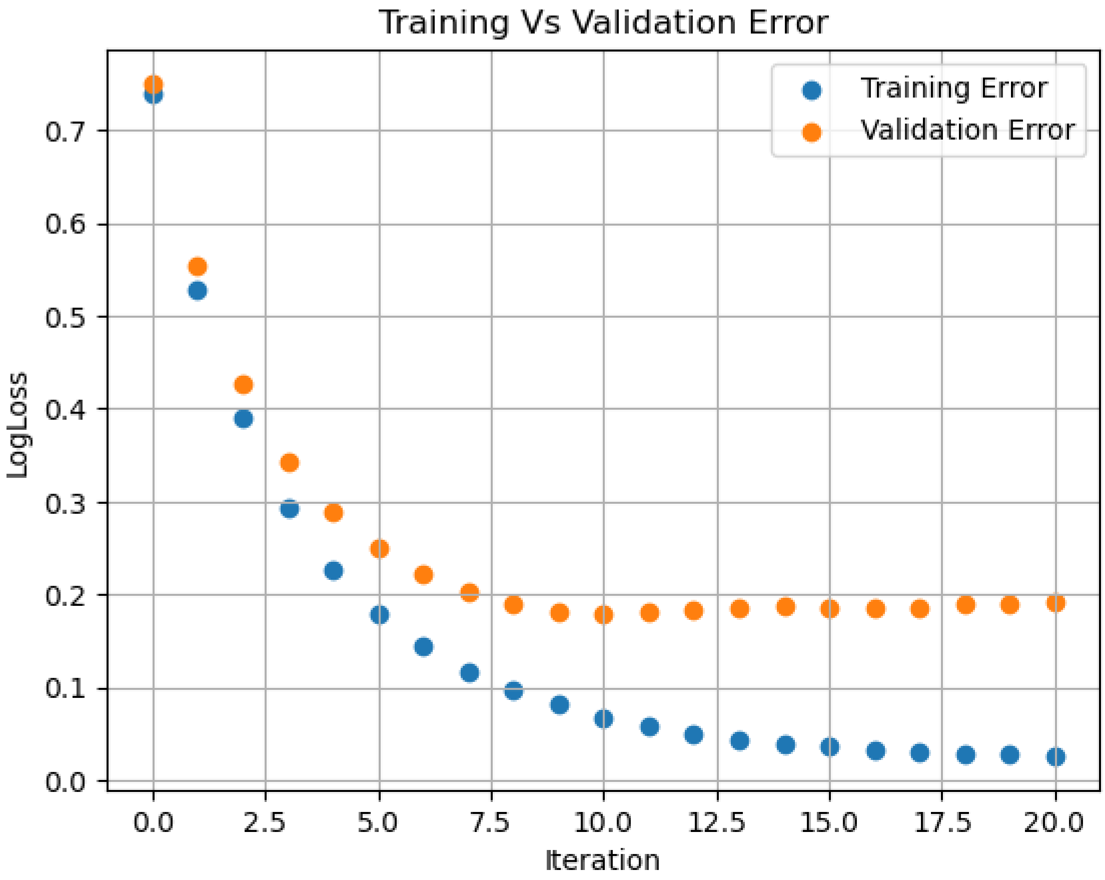
In [19]:

```
print(training_rounds)
```

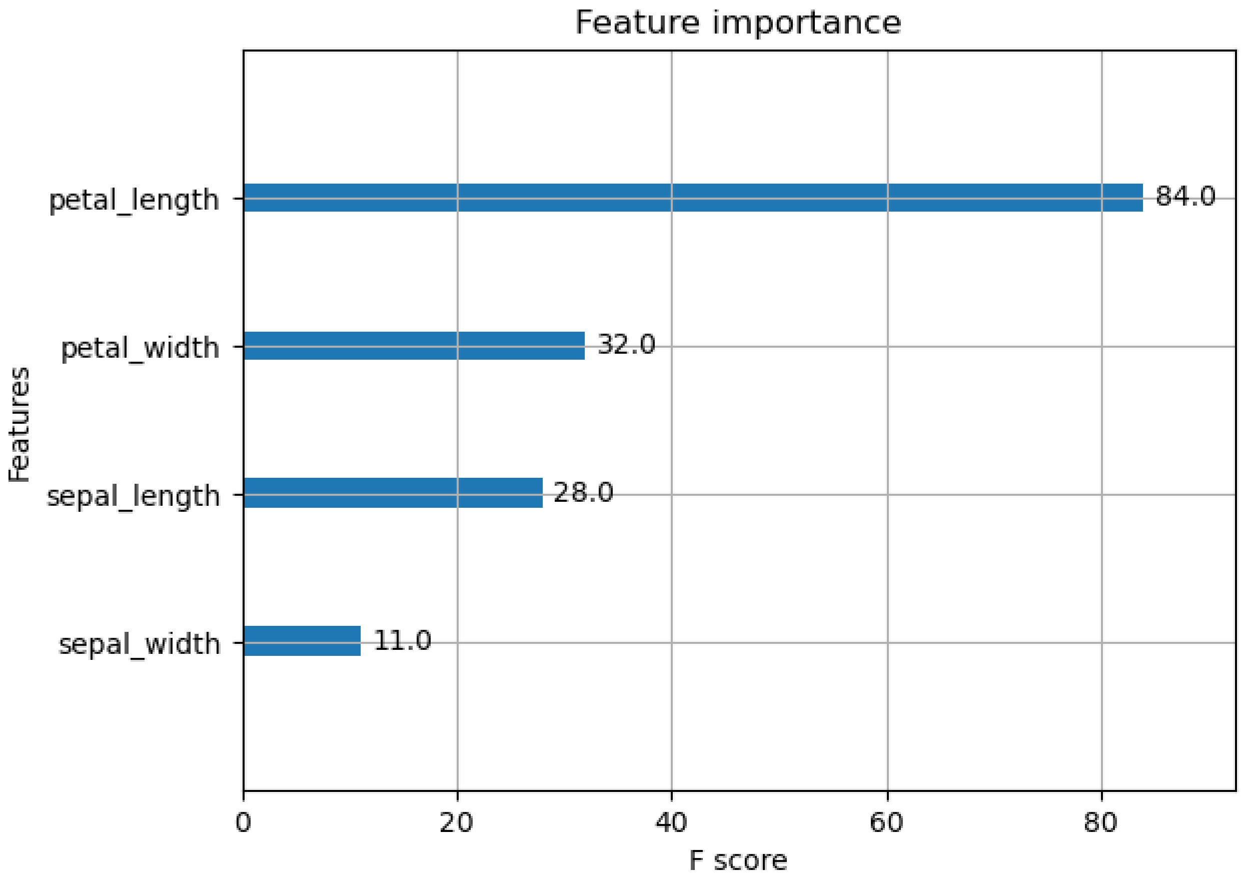
```
range(0, 21)
```

In [20]:

```
plt.scatter(x=training_rounds,y=eval_result['validation_0']['mlogloss'],label='Training Error')
plt.scatter(x=training_rounds,y=eval_result['validation_1']['mlogloss'],label='Validation Error')
plt.grid(True)
plt.xlabel('Iteration')
plt.ylabel('LogLoss')
plt.title('Training Vs Validation Error')
plt.legend()
plt.show()
```



```
In [21]: xgb.plot_importance(classifier)
         plt.show()
```



In [22]:
```
df = pd.read_csv(validation_file,names=columns)
```

In [23]:
```
df.head()
```

Out[23]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width
0	1	5.8	2.7	4.1	1.0
1	0	4.8	3.4	1.6	0.2
2	1	6.0	2.2	4.0	1.0
3	2	6.4	3.1	5.5	1.8
4	2	6.7	2.5	5.8	1.8

In [24]:

```
X_test = df.iloc[:,1:]
print(X_test[:5])
```

```
   sepal_length  sepal_width  petal_length  petal_width
0           5.8          2.7           4.1          1.0
1           4.8          3.4           1.6          0.2
2           6.0          2.2           4.0          1.0
3           6.4          3.1           5.5          1.8
4           6.7          2.5           5.8          1.8
```

In [25]:

```
result = classifier.predict(X_test)
```

In [26]:

```
result[:5]
```

Out[26]:

```
array([1, 0, 1, 2, 2], dtype=int32)
```

In [27]:

```
df['predicted_class'] = result #le.inverse_transform(result)
#DWB#                      = le.inverse_transform(result) #DWB# to get class names
```

In [28]:

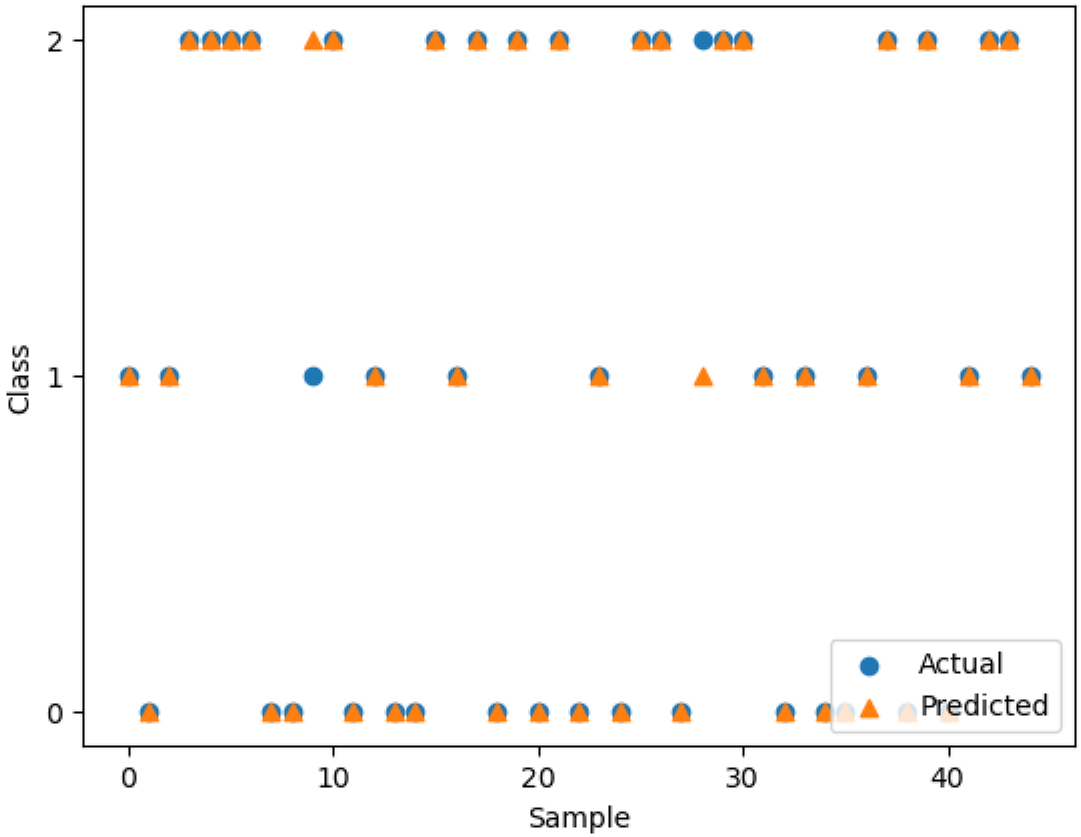
```
df.head()
```

Out[28]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width	predicted_class
0	1	5.8	2.7	4.1	1.0	1
1	0	4.8	3.4	1.6	0.2	0
2	1	6.0	2.2	4.0	1.0	1
3	2	6.4	3.1	5.5	1.8	2
4	2	6.7	2.5	5.8	1.8	2

In [29]:

```
# Compare performance of Actual and Model 1 Prediction
plt.figure()
plt.scatter(df.index,df['encoded_class'],label='Actual')
plt.scatter(df.index,df['predicted_class'],label='Predicted',marker='^')
plt.legend(loc=4)
plt.yticks([0,1,2])
plt.xlabel('Sample')
plt.ylabel('Class')
plt.show()
```



```
In [30]: #DWB#  After I make a copy of this DataFrame that Chandra had
         #DWB#+ in the notebook
         df_orig = df.copy(deep=True)

         #DWB# Doing what was commented
         df['pred_cls_decoded'] = le.inverse_transform(result)
         df['decoded_cls'] = le.inverse_transform(df['encoded_class'])
```

```
In [31]: #DWB# Seeing result of doing what was commented
         df.head()
```

Out[31]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width	predicted_class	pred_cls_decoded	decoded_cls
0	1	5.8	2.7	4.1	1.0	1	Iris-versicolor	Iris-versicolor
1	0	4.8	3.4	1.6	0.2	0	Iris-setosa	Iris-setosa
2	1	6.0	2.2	4.0	1.0	1	Iris-versicolor	Iris-versicolor
3	2	6.4	3.1	5.5	1.8	2	Iris-virginica	Iris-virginica
4	2	6.7	2.5	5.8	1.8	2	Iris-virginica	Iris-virginica

In [32]:

```
print(df.head())
```

```
   encoded_class  sepal_length  sepal_width  petal_length  petal_width  
0              1           5.8          2.7           4.1          1.0  \
1              0           4.8          3.4           1.6          0.2   
2              1           6.0          2.2           4.0          1.0   
3              2           6.4          3.1           5.5          1.8   
4              2           6.7          2.5           5.8          1.8   

   predicted_class pred_cls_decoded      decoded_cls  
0                1  Iris-versicolor  Iris-versicolor  
1                0      Iris-setosa      Iris-setosa  
2                1  Iris-versicolor  Iris-versicolor  
3                2   Iris-virginica   Iris-virginica  
4                2   Iris-virginica   Iris-virginica  
```

In [33]:

```
#DWB# Getting the data back to normal for the confusion matrix
df['pred_cls_enc'] = result
df['enc_cls_chk'] = le.transform(df['decoded_cls'])
```

In [34]:

```
#DWB#  Checking it worked.
df.head()
```

Out[34]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width	predicted_class	pred_cls_decoded	decoded_cls	pred_cls_enc	enc_cl
0	1	5.8	2.7	4.1	1.0	1	Iris-versicolor	Iris-versicolor	1	
1	0	4.8	3.4	1.6	0.2	0	Iris-setosa	Iris-setosa	0	
2	1	6.0	2.2	4.0	1.0	1	Iris-versicolor	Iris-versicolor	1	
3	2	6.4	3.1	5.5	1.8	2	Iris-virginica	Iris-virginica	2	
4	2	6.7	2.5	5.8	1.8	2	Iris-virginica	Iris-virginica	2	

In [35]:

```
print(df.head())
```

```
   encoded_class  sepal_length  sepal_width  petal_length  petal_width  \
0              1           5.8          2.7           4.1          1.0   
1              0           4.8          3.4           1.6          0.2   
2              1           6.0          2.2           4.0          1.0   
3              2           6.4          3.1           5.5          1.8   
4              2           6.7          2.5           5.8          1.8   

   predicted_class pred_cls_decoded      decoded_cls  pred_cls_enc  \
0                1  Iris-versicolor  Iris-versicolor             1   
1                0      Iris-setosa      Iris-setosa             0   
2                1  Iris-versicolor  Iris-versicolor             1   
3                2   Iris-virginica   Iris-virginica             2   
4                2   Iris-virginica   Iris-virginica             2   

   enc_cls_chk  
0            1  
1            0  
2            1  
3            2  
4            2  
```

In [36]:

```
#DWB#  I will actually get rid of those extra columns
#DWB#+ before starting the next section.
df.drop(columns=['pred_cls_decoded', 'decoded_cls',
                 'pred_cls_enc', 'enc_cls_chk'], inplace=True)
```

In [37]:

```python
# Checking result
df.head()
```

Out[37]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width	predicted_class
0	1	5.8	2.7	4.1	1.0	1
1	0	4.8	3.4	1.6	0.2	0
2	1	6.0	2.2	4.0	1.0	1
3	2	6.4	3.1	5.5	1.8	2
4	2	6.7	2.5	5.8	1.8	2

In [38]:

```python
# Quick, non-thorough check that we're back
df_orig.head()
```

Out[38]:

	encoded_class	sepal_length	sepal_width	petal_length	petal_width	predicted_class
0	1	5.8	2.7	4.1	1.0	1
1	0	4.8	3.4	1.6	0.2	0
2	1	6.0	2.2	4.0	1.0	1
3	2	6.4	3.1	5.5	1.8	2
4	2	6.7	2.5	5.8	1.8	2

Confusion Matrix

Confusion Matrix is a table that summarizes performance of classification model.

In [39]:

```python
# Reference:
# https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
```

```
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        #print("Normalized confusion matrix")
    #else:
    #    print('Confusion matrix, without normalization')

    #print(cm)

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, format(cm[i, j], fmt),
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")

    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

In [40]:

```
# Compute confusion matrix
cnf_matrix = confusion_matrix(df['encoded_class'],
                              df['predicted_class'],labels=labels)
```

In [41]:

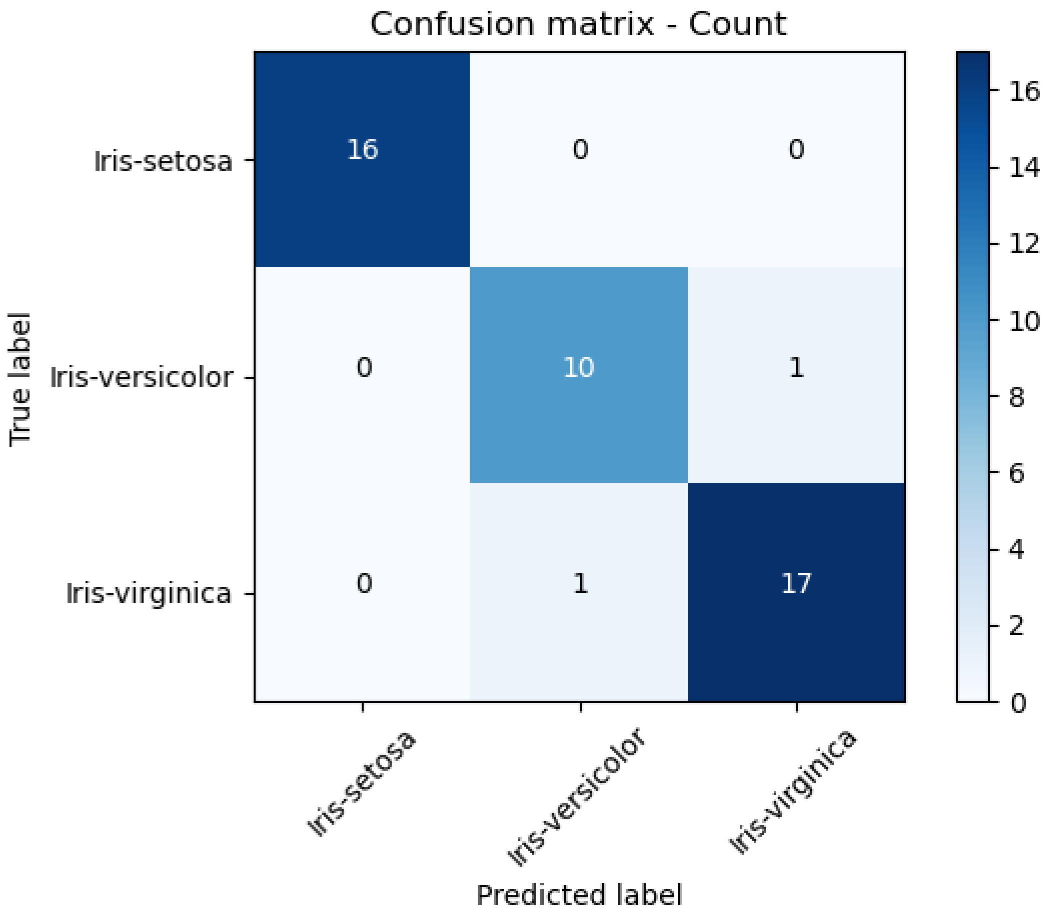
```
cnf_matrix
```

Out[41]:

```
array([[16,  0,  0],
       [ 0, 10,  1],
       [ 0,  1, 17]])
```

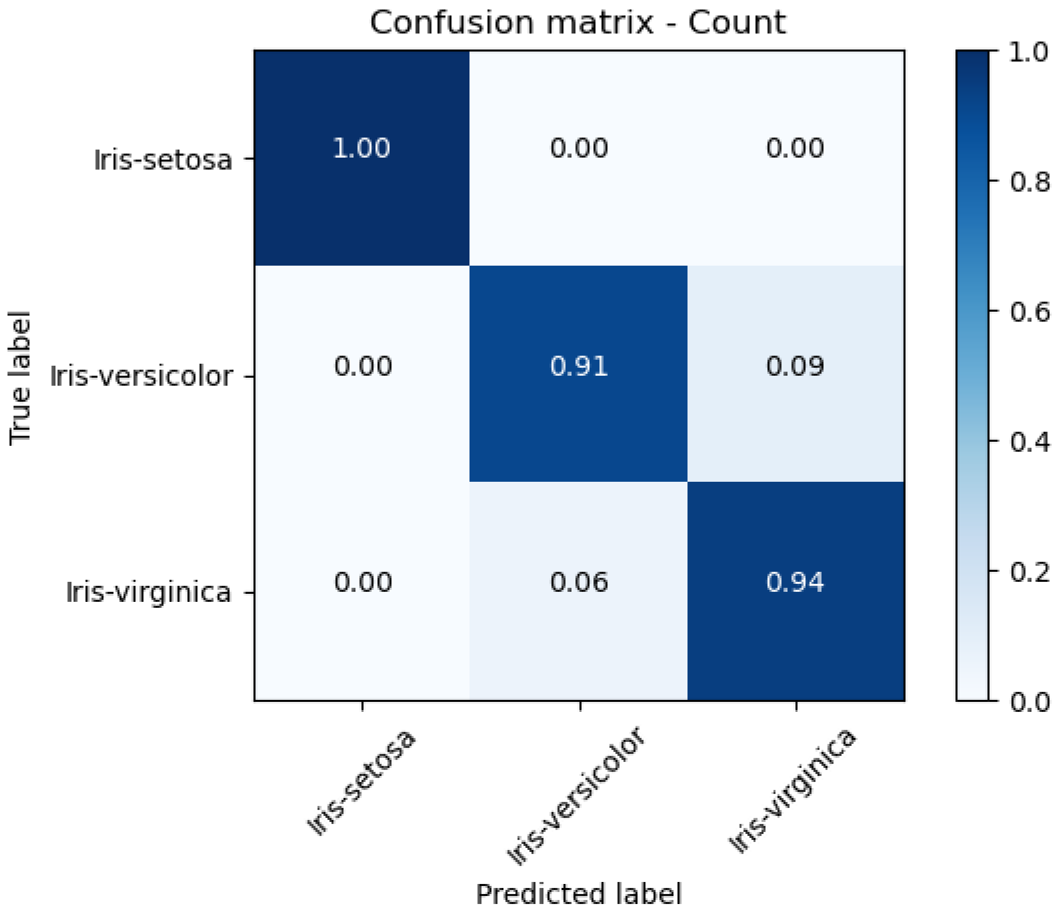
In [42]:

```
# Plot confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=classes,
                      title='Confusion matrix - Count')
```



In [43]:

```
# Plot confusion matrix
plt.figure()
plot_confusion_matrix(cnf_matrix, classes=classes,
                      title='Confusion matrix - Count',normalize=True)
```



In [ ]:

```
#DWB#  Note that, with this being a multi-class classification problem,
#DWB#+ weighted average f1-score is a good metric.
print(classification_report(df['encoded_class'],
                            df['predicted_class'],
                            labels=labels,
                            target_names=classes))
```

In [ ]: