In [1]:
```
#Importing libraries
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [29]:
```
# Loading the Iris dataset
iris = load_iris()

# Accessing the features (input variables)
features = iris.data
print("Features:\n", features)
```

```
[6.  2.2 5.  1.5]
[6.9 3.2 5.7 2.3]
[5.6 2.8 4.9 2. ]
[7.7 2.8 6.7 2. ]
[6.3 2.7 4.9 1.8]
[6.7 3.3 5.7 2.1]
[7.2 3.2 6.  1.8]
[6.2 2.8 4.8 1.8]
[6.1 3.  4.9 1.8]
[6.4 2.8 5.6 2.1]
[7.2 3.  5.8 1.6]
[7.4 2.8 6.1 1.9]
[7.9 3.8 6.4 2. ]
[6.4 2.8 5.6 2.2]
[6.3 2.8 5.1 1.5]
[6.1 2.6 5.6 1.4]
[7.7 3.  6.1 2.3]
[6.3 3.4 5.6 2.4]
```

In [30]:
```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
scaler.fit(features)
scaled_features=scaler.transform(features)
```

In [31]:
```
# Accessing the target values (output variable)
target = iris.target
print("Target:\n", target)
```

```
Target:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
```

In [32]:
```
# Getting the column names for independent variables (features)
feature_names = iris.feature_names
print("Independent variable (feature) column names:")
print(feature_names)

# Getting the column name for the dependent variable (target)
target_name = iris.target_names
print("Dependent variable (target) column name:")
print(target_name)
```

```
Independent variable (feature) column names:
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
Dependent variable (target) column name:
['setosa' 'versicolor' 'virginica']
```

In [34]:
```
#Splitting the data
from sklearn.model_selection import train_test_split
# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Displaying the shapes of the resulting datasets
print("Training set shape - X:", X_train.shape, "y:", y_train.shape)
print("Testing set shape - X:", X_test.shape, "y:", y_test.shape)
```

```
Training set shape - X: (120, 4) y: (120,)
Testing set shape - X: (30, 4) y: (30,)
```

In [35]:
```
from sklearn.model_selection import KFold
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import ElasticNet
from sklearn.svm import SVC


from sklearn.model_selection import cross_val_score
kfold = KFold(n_splits=10, shuffle=True)

KNC = KNeighborsClassifier()
DTC = DecisionTreeClassifier()
LR = LogisticRegression(max_iter=300)
RFC = RandomForestClassifier()
EN = ElasticNet()


models = [KNC, DTC, LR, RFC, EN]

for i in models:
  model_score = cross_val_score(i,features,target,cv=kfold)
  print(f"Cross validation scores for {i}= {model_score}")
  print(f"Model Accuracy for {i}={model_score.mean()}")
  print('')
```

```
Cross validation scores for KNeighborsClassifier()= [1.         0.93333333 1.         0.93333333 0.93333333 0.93333333
 1.         0.93333333 1.         1.        ]
Model Accuracy for KNeighborsClassifier()=0.9666666666666666

Cross validation scores for DecisionTreeClassifier()= [1.         0.8        0.93333333 0.93333333 0.86666667 1.
 0.8        1.         1.         1.        ]
Model Accuracy for DecisionTreeClassifier()=0.9333333333333333

Cross validation scores for LogisticRegression(max_iter=300)= [1.         1.         1.         1.         1.         0.9333
3333
 0.8        1.         1.         0.93333333]
Model Accuracy for LogisticRegression(max_iter=300)=0.9666666666666668

Cross validation scores for RandomForestClassifier()= [0.93333333 1.         0.86666667 1.         1.         0.86666667
 1.         1.         0.93333333 1.        ]
Model Accuracy for RandomForestClassifier()=0.96

Cross validation scores for ElasticNet()= [0.74864287 0.72293289 0.51223733 0.63833465 0.75848265 0.68838167
 0.6582322  0.69535868 0.7358627  0.69436966]
Model Accuracy for ElasticNet()=0.6852835299512695
```

In [44]:
```
#Choosing the algorithm
from sklearn.ensemble import RandomForestClassifier
RFC = RandomForestClassifier()
RFC.fit(X_train,y_train)
```

Out[44]:
```
RandomForestClassifier()
```

In [45]:
```
y_pred = RFC.predict(X_test)
y_pred
```

Out[45]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

In [47]:
```
from sklearn.metrics import accuracy_score, precision_score, recall_score
#finding metrics
def get_metrics(y_true, y_pred):
    accuracy = accuracy_score(y_true, y_pred)
    precision = precision_score(y_true, y_pred, average='macro')
    recall = recall_score(y_true, y_pred, average='macro')

    return {'accuracy' : round(accuracy, 2), 'precision' : round(precision, 2), 'recall' : round(recall, 2)}
```

In [48]:
```
# Calculating the metrics using the get_metrics function
run_metrics = get_metrics(y_test, y_pred)

print(run_metrics)
```

```
{'accuracy': 1.0, 'precision': 1.0, 'recall': 1.0}
```

In [50]:
```
import mlflow
```

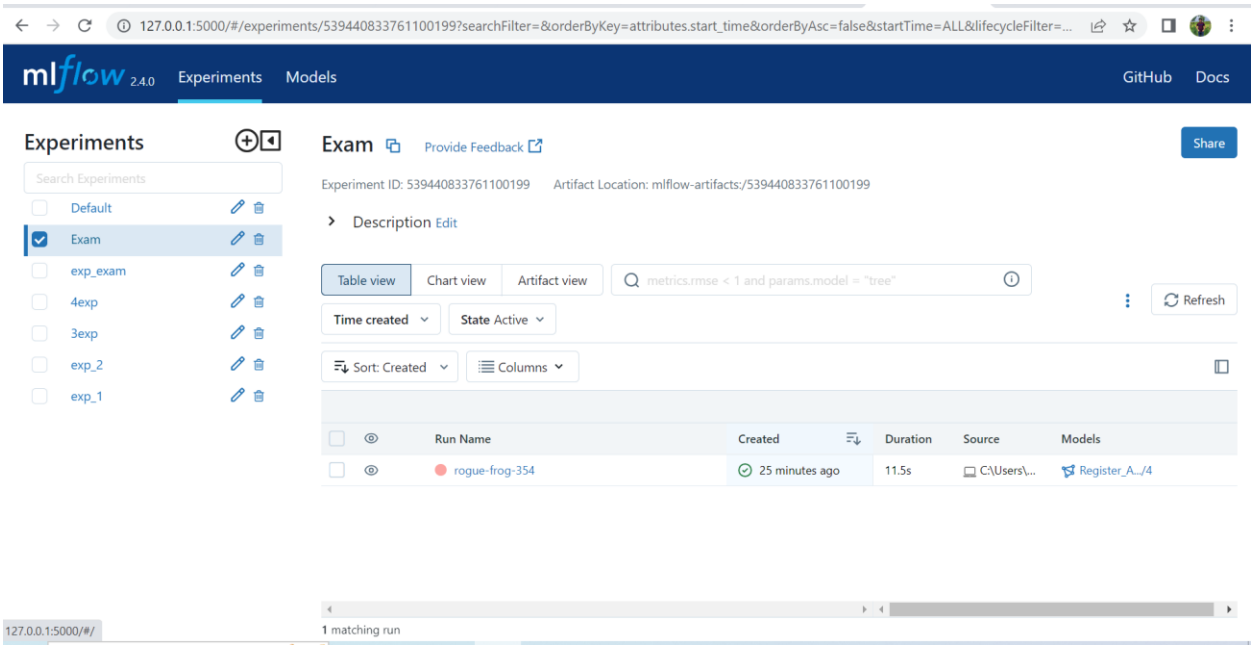
In [64]:
```
mlflow.set_tracking_uri("http://localhost:5000")
mlflow.set_experiment(experiment_name)
with mlflow.start_run(run_name="Exam"):
        for metric in run_metrics:
            mlflow.log_metric(metric,run_metrics[metric])

        #send the model also
        mlflow.sklearn.log_model(RFC,"model", registered_model_name= 'Register_Assesment_Model_Exam')

print("It is logged to Experiment - %s" %(1))
```

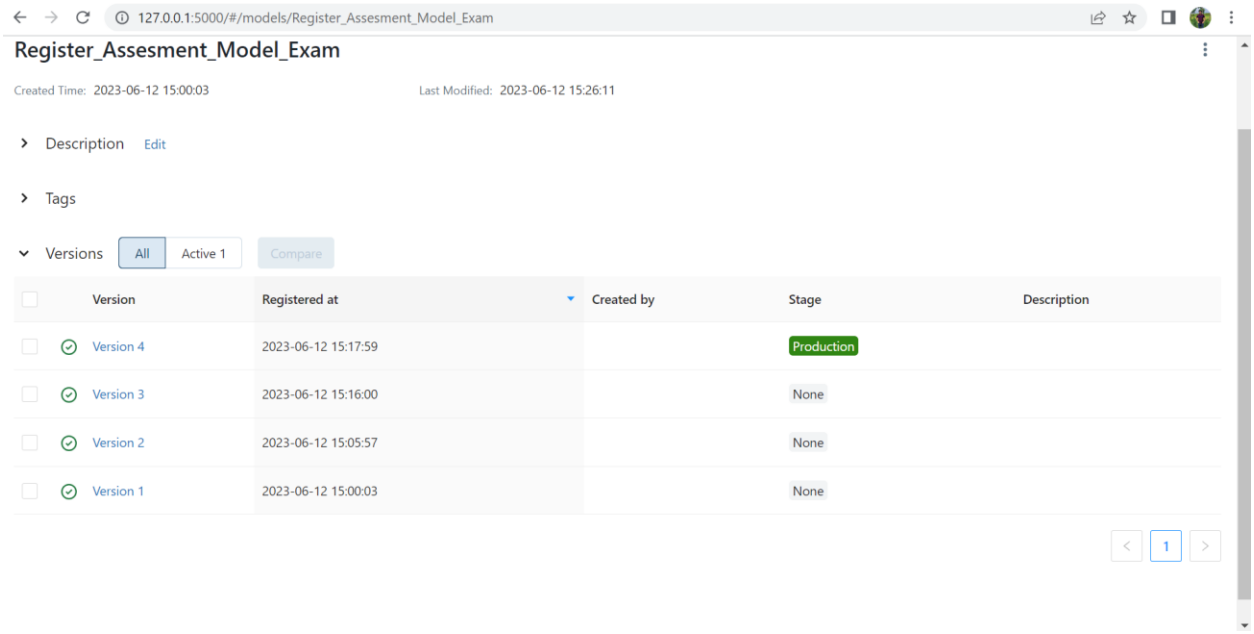
```
Registered model 'Register_Assesment_Model_Exam' already exists. Creating a new version of this model...
2023/06/12 15:16:00 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version to finish creat
ion. Model name: Register_Assesment_Model_Exam, version 3
```

```
It is logged to Experiment - 1
```

```
Created version '3' of model 'Register_Assesment_Model_Exam'.
```

From Mlflow UI:



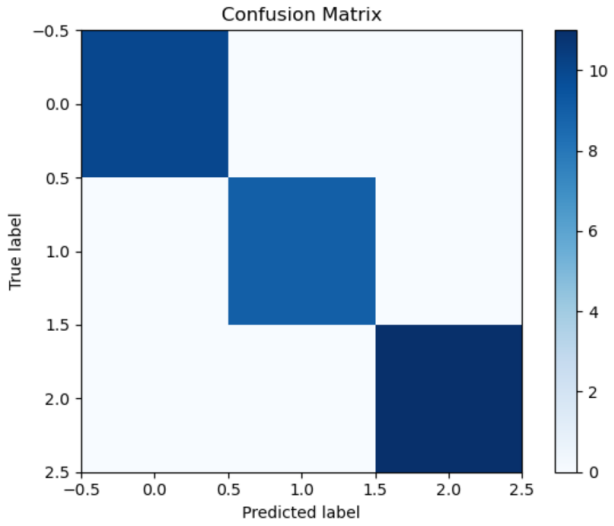


In [65]:
```
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

def create_confusion_matrix_plot(knn, X_test, y_test):
    # Assuming you have your predicted labels in y_pred
    y_pred = RFC.predict(X_test)

    # Creating the confusion matrix
    cm = confusion_matrix(y_test, y_pred)

    # Plotting the confusion matrix
    plt.imshow(cm, interpolation='nearest', cmap=plt.cm.Blues)
    plt.title('Confusion Matrix')
    plt.colorbar()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
    plt.tight_layout()
```

In [66]:
```
# Saving the graph
plt.savefig('confusion_matrix.png')
plt.show()
```

```
<Figure size 640x480 with 0 Axes>
```

In [67]:
```
create_confusion_matrix_plot(RFC, X_test, y_test)
```



In [74]:

```
import mlflow

#Setting the experiment
mlflow.set_tracking_uri("http://localhost:5000")
mlflow.set_experiment("exp_exam")

#Starting the mlflow
with mlflow.start_run():
    #Check if metrics present in this run_metrics if yes log that
    get_metrics = ['accuracy', 'precision', 'recall']
    res = {'accuracy': 1.0, 'precision': 1.0, 'recall': 1.0}
    for metric in get_metrics:
            mlflow.log_metric(metric, res[metric])


    #We use artifact when we are saving files
    mlflow.log_artifact('confusion_matrix1.png', 'confusion_materix')

    #This below code we use for tag
    mlflow.set_tag("tag1", "RFC Algo")
    mlflow.set_tags({"tag2":"Testing2", "tag3":"Testing3", "tag4":"Testing4"})

    #sending the model also
    mlflow.sklearn.log_model(RFC,"model", registered_model_name= 'Register_Assesment_Model_Exam')
```

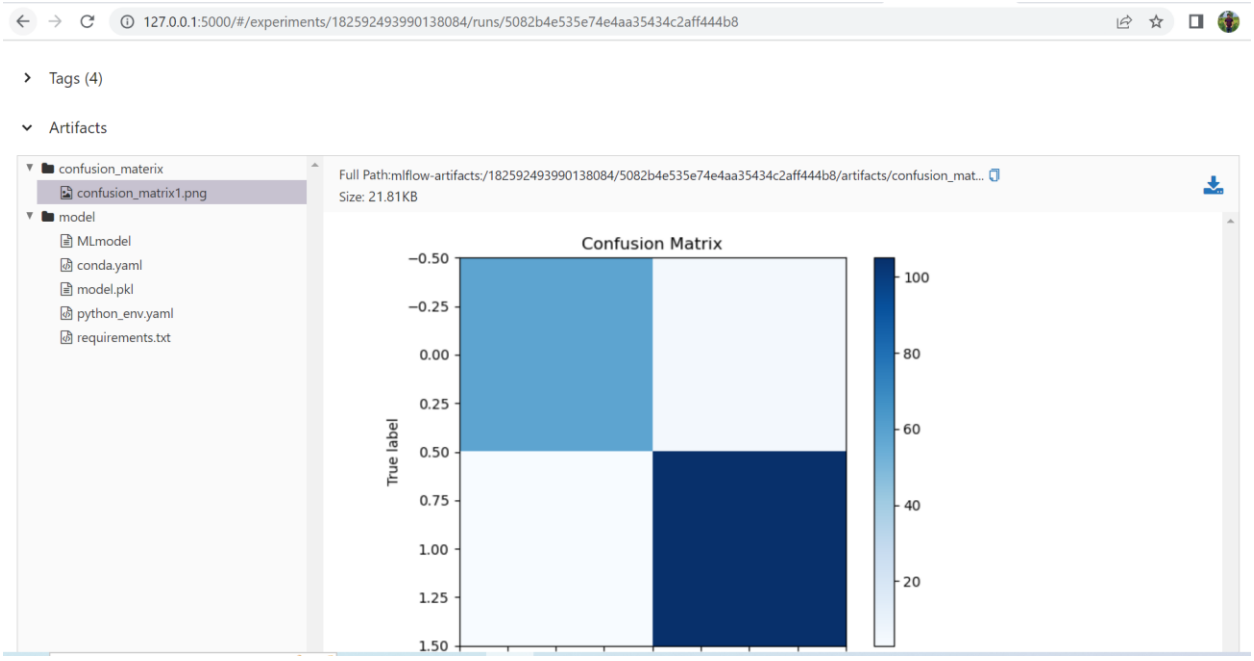
```
print('It is logged to Experiment - %s' %( experiment_name))
```

```
2023/06/12 15:17:47 INFO mlflow.tracking.fluent: Experiment with name 'Exam' does not exist. Creating a new experiment.
Registered model 'Register_Assesment_Model_Exam' already exists. Creating a new version of this model...
2023/06/12 15:17:59 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version to finish creat
ion. Model name: Register_Assesment_Model_Exam, version 4
```

```
It is logged to Experiment - exp_exam
```

```
Created version '4' of model 'Register_Assesment_Model_Exam'.
```

From Mlflow UI:



In [70]:
```
#for question 2
# Predicting on a Pandas DataFrame:
import mlflow
logged_model = 'runs:/49c5689f09c14b0598115fcc24a393ff/model'

# Loading model as a PyFuncModel.
loaded_model = mlflow.pyfunc.load_model(logged_model)

# Predicting on a Pandas DataFrame.
import pandas as pd
loaded_model.predict(pd.DataFrame(X_test))
```

Out[70]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

In [71]:
```
import mlflow.pyfunc

model_name = "Register_Assesment_Model_Exam"
model_version = 4

model = mlflow.pyfunc.load_model(model_uri=f"models:/{model_name}/{model_version}")

y_pred = model.predict(X_test)
y_pred
```

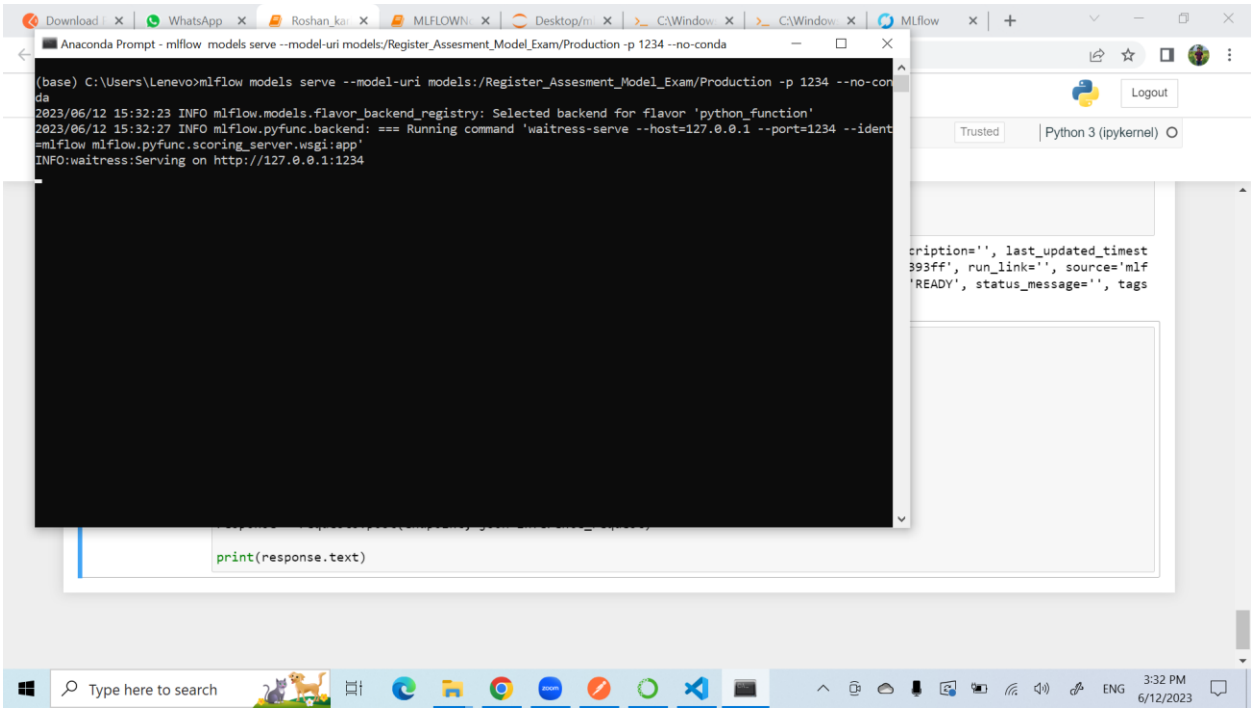
Out[71]:
```
array([1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2,
       0, 2, 2, 2, 2, 2, 0, 0])
```

In [72]:
```
#Registering this model to production stage
client = mlflow.tracking.MlflowClient()
client.transition_model_version_stage(
name= "Register_Assesment_Model_Exam",
version=4,
stage="Production")
```

Out[72]:
```
<ModelVersion: aliases=[], creation_timestamp=1686562379240, current_stage='Production', description='', last_updated_timest
amp=1686562871561, name='Register_Assesment_Model_Exam', run_id='49c5689f09c14b0598115fcc24a393ff', run_link='', source='mlf
low-artifacts:/539440833761100199/49c5689f09c14b0598115fcc24a393ff/artifacts/model', status='READY', status_message='', tags
={}, user_id='', version='4'>
```

In [77]:
```
# Creating a REST API locally with MLflow serving
# -Using model
import requests
import json

inference_request = {
       "dataframe_records":pd.DataFrame(X_test).values.tolist()
}

endpoint = "http://localhost:1234/invocations"

response = requests.post(endpoint, json=inference_request)

print(response.text)
```

```
{"predictions": [1, 0, 2, 1, 1, 0, 1, 2, 1, 1, 2, 0, 0, 0, 0, 1, 2, 1, 1, 2, 0, 2, 0, 2, 2, 2, 2, 2, 0, 0]}
```


Anaconda Prompt - mlflow models serve --model-uri models:/Register_Assesment_Model_Exam/Production -p 1234 --no-conda
(base) C:\Users\Lenevo>mlflow models serve --model-uri models:/Register_Assesment_Model_Exam/Production -p 1234 --no-conda
2023/06/12 15:32:23 INFO mlflow.models.flavor_backend_registry: Selected backend for flavor 'python_function'
2023/06/12 15:32:27 INFO mlflow.pyfunc.backend: === Running command 'waitress-serve --host=127.0.0.1 --port=1234 --ident=mlflow mlflow.pyfunc.scoring_server.wsgi:app'
INFO:waitress:Serving on http://127.0.0.1:1234
Logout
Trusted
Python 3 (ipykernel)
print(response.text)
Type here to search
ENG
3:32 PM
6/12/2023