6. Write a program to implement K-Nearest Neighbour algorithm to classify the iris data set. Print both correct and wrong predictions. Java/Python ML library classes can be used for this problem.

```
from sklearn.datasets import load_iris
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
iris_dataset = load_iris()


print("\n IRIS FEATURES \ TARGET NAMES:\n ", iris_dataset.
    target_names)
for i in range (len(iris_dataset.target_names)):
    print("\n {0}]: [{1}] ".format(i, iris_dataset.target_
        names[i]))
#print("\n IRIS DATA:\n ", iris_dataset["data"])


x_train, x_test, y_train, y_test = train_test_split (iris_dataset
["data"], iris_dataset["target"], random_state = 0)


classifier = KNeighborsClassifier(n_neighbors = 8, p=3, metric=
                    'euclidean')
classifier.fit(x_train, y_train)


y_pred = classifier.predict (x_test)
```

ARUN'S

Scanned with CamScanner

```python
cm = confusion_matrix(y_test, y_pred)
print('Confusion matrix is as follows\n', cm)
print('Accuracy metrics')
print(classification_report(y_test, y_pred))
print("Correct prediction", accuracy_score(y_test, y_pred))
print("Wrong prediction", (1- accuracy_score(y_test, y_pred)))
```

Output:

IRIS FEATURES \ TARGET NAMES:

['setosa' 'versicolor' 'virginica']

[0] : [setosa]

[1] : [versicolor]

[2] : [virginica]

KNeighbos Classifier(algorithm = 'auto', leaf_size = 30, metric = 'euclidear

metric_param = None, n_jobs = None, n_neighbos:8

P=3, weight = 'uniform')

Confusion matrix is as follows

[[13 0 0]

 [0 15 1]

 [0 0 9]]

Accuracy metrics

	precision	recall	f1-score	support
0	1.00	1.00	1.00	13
1	1.00	0.94	0.97	6
2	0.90	1.00	0.95	9
accuracy			0.97	38
macro avg	0.97	0.98	0.97	38
weighted avg	0.98	0.97	0.97	38

Correct prediction 0.9736842105263158

Wrong prediction 0.02631578947368418