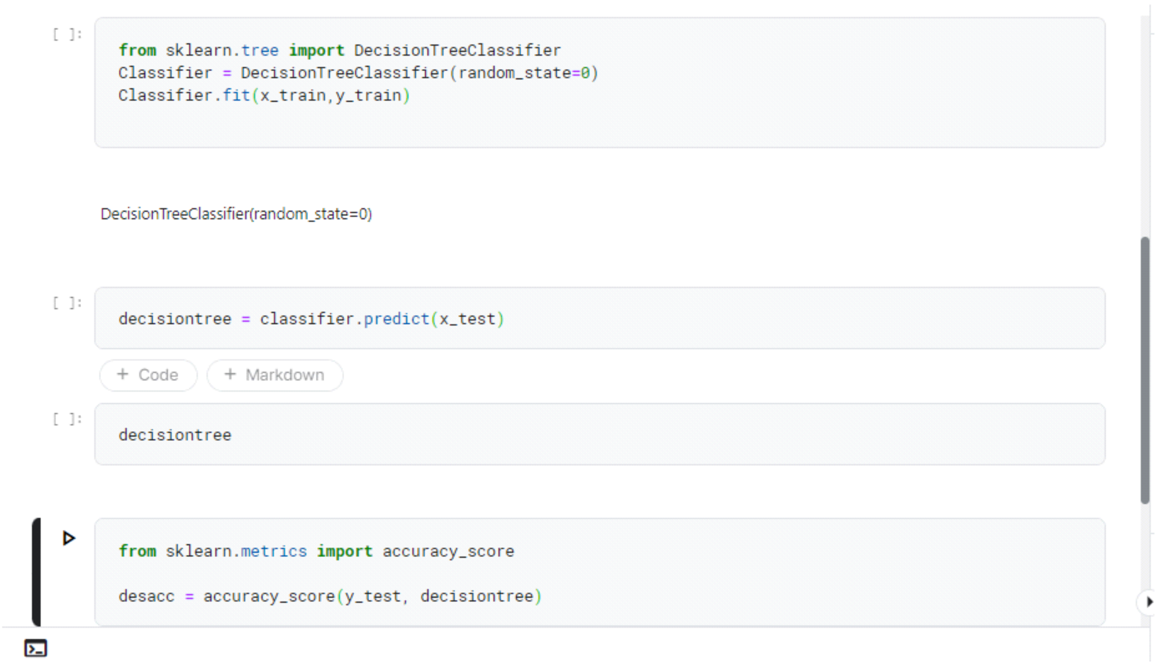
# **NM2023TMID32012 -** Flight Delay Prediction

# Milestone 4: Model Building

## Activity 1: Training the model in multiple algorithms

### Activity 1.1: Decision tree model



```
[ ]: from sklearn.tree import DecisionTreeClassifier
     Classifier = DecisionTreeClassifier(random_state=0)
     Classifier.fit(x_train,y_train)
```

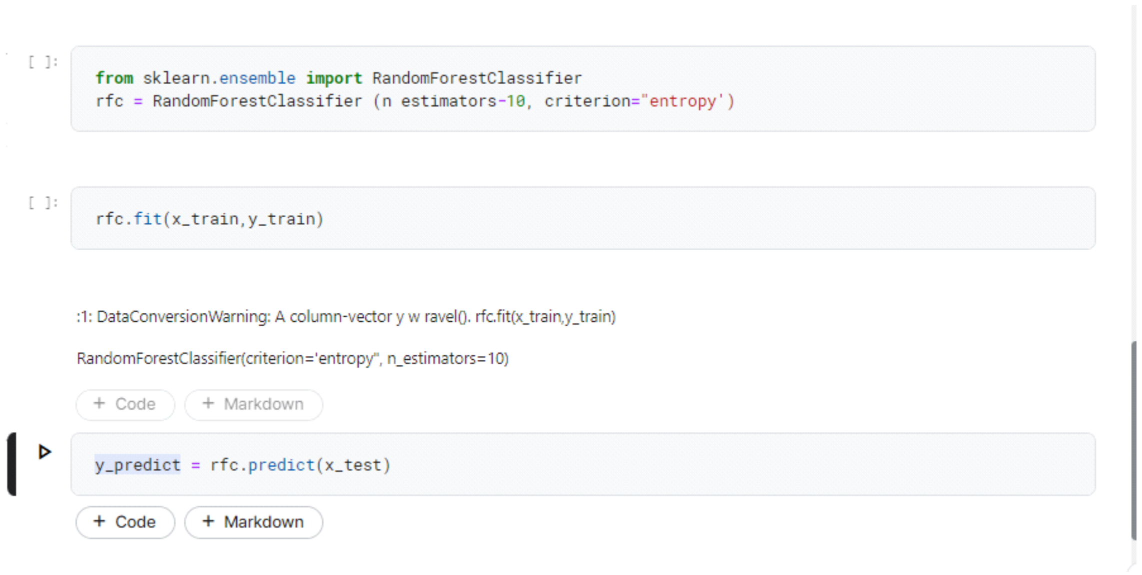
```
DecisionTreeClassifier(random_state=0)
```

```
[ ]: decisiontree = classifier.predict(x_test)
```

```
[ ]: decisiontree
```

```
from sklearn.metrics import accuracy_score

desacc = accuracy_score(y_test, decisiontree)
```

### Activity 1.2: Random forest model



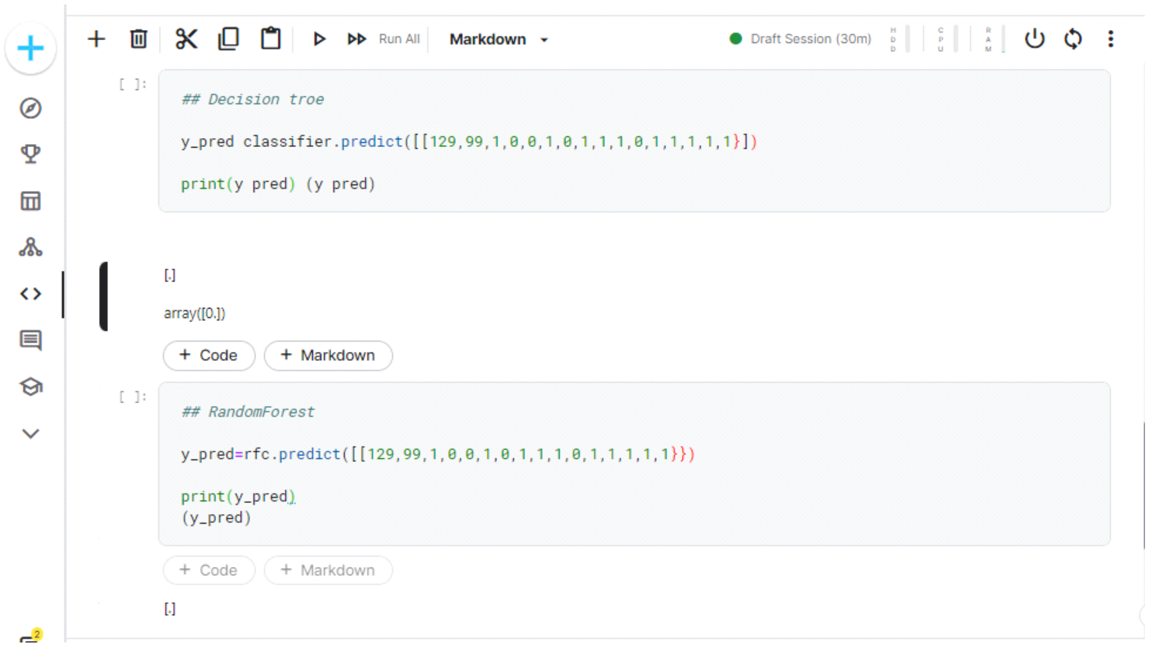
```
[ ]: from sklearn.ensemble import RandomForestClassifier
     rfc = RandomForestClassifier (n estimators-10, criterion="entropy')
```

```
[ ]: rfc.fit(x_train,y_train)
```

```
:1: DataConversionWarning: A column-vector y w ravel(). rfc.fit(x_train,y_train)

RandomForestClassifier(criterion='entropy", n_estimators=10)
```

```
y_predict = rfc.predict(x_test)
```

## Activity 2: Test the model

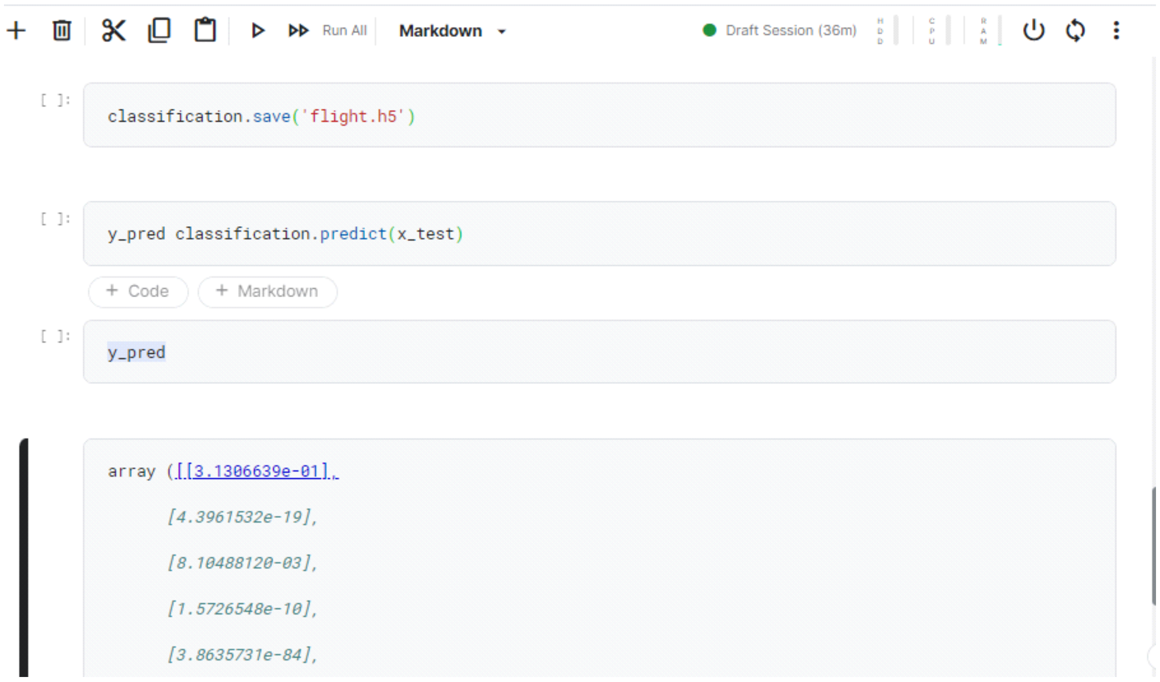


```
[ ]:
## Decision troe

y_pred classifier.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1}])

print(y pred) (y pred)
```

```
[.]

array([0.])
```

```
[ ]:
## RandomForest

y_pred=rfc.predict([[129,99,1,0,0,1,0,1,1,1,0,1,1,1,1,1}})

print(y_pred)
(y_pred)
```

```
[.]
```

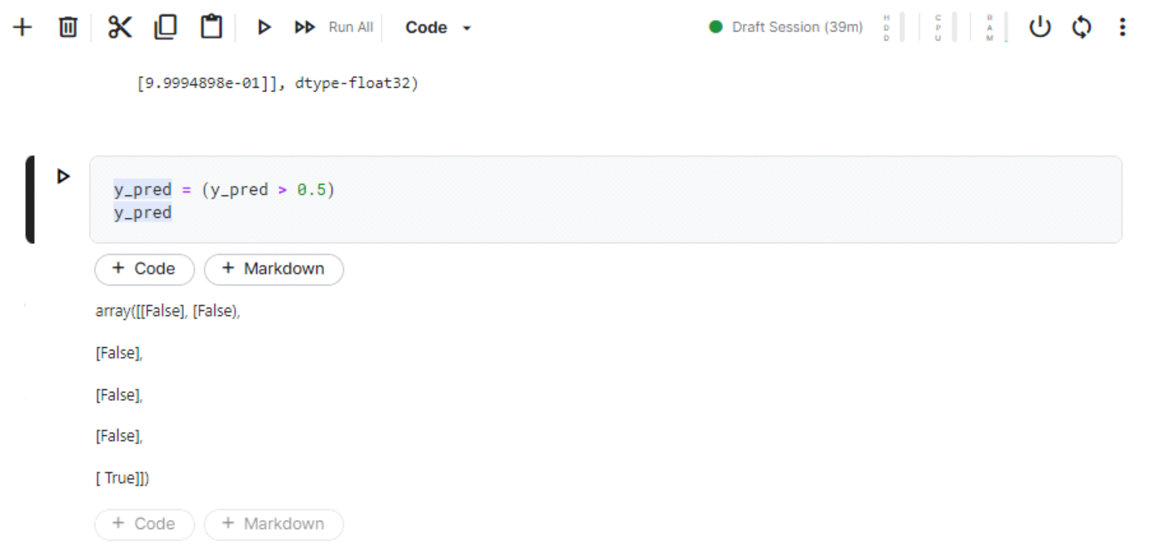


```
[ ]: classification.save('flight.h5')
```

```
[ ]: y_pred classification.predict(x_test)
```

```
[ ]: y_pred
```

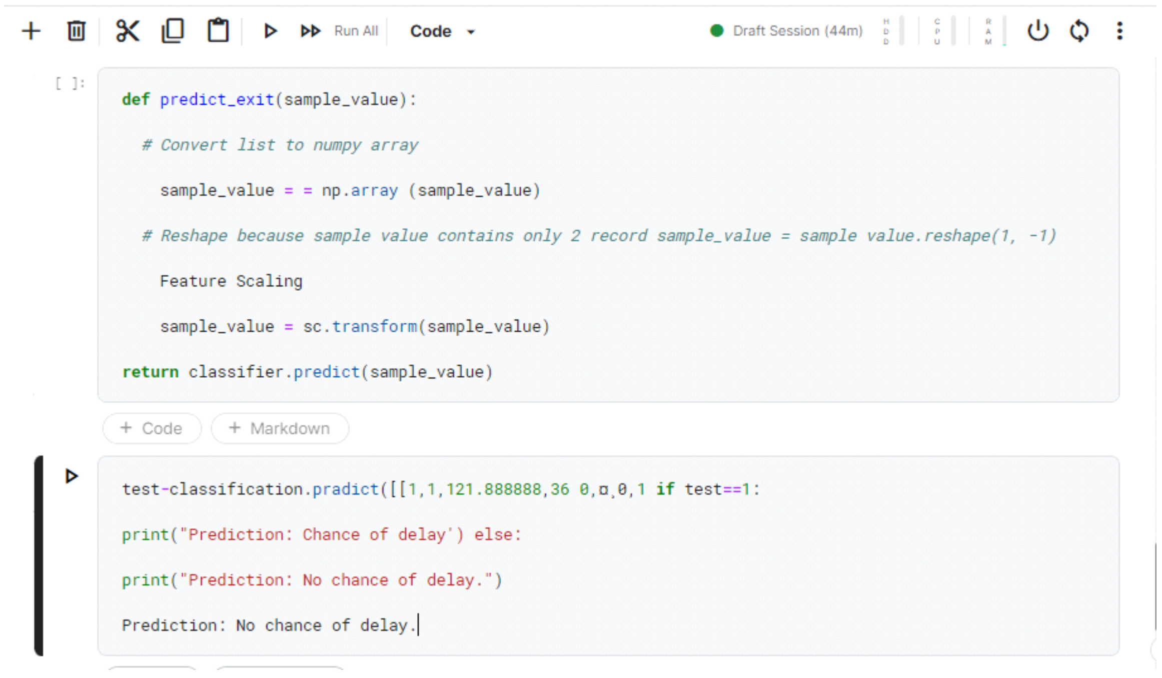
```
array ([[3.1306639e-01],

       [4.3961532e-19],

       [8.10488120-03],

       [1.5726548e-10],

       [3.8635731e-04],
```



```
[9.9994898e-01]], dtype-float32)
```

```
y_pred = (y_pred > 0.5)
y_pred
```

```
array([[False], [False),

[False],

[False],

[False],

[ True]])
```



```
def predict_exit(sample_value):

  # Convert list to numpy array

    sample_value = = np.array (sample_value)

  # Reshape because sample value contains only 2 record sample_value = sample value.reshape(1, -1)

    Feature Scaling

    sample_value = sc.transform(sample_value)

return classifier.predict(sample_value)
```

```
test-classification.pradict([[1,1,121.888888,36 0,¤,0,1 if test==1:

print("Prediction: Chance of delay') else:

print("Prediction: No chance of delay.")

Prediction: No chance of delay.
```