Pycaret Assignment

❖ Multiclass Classification

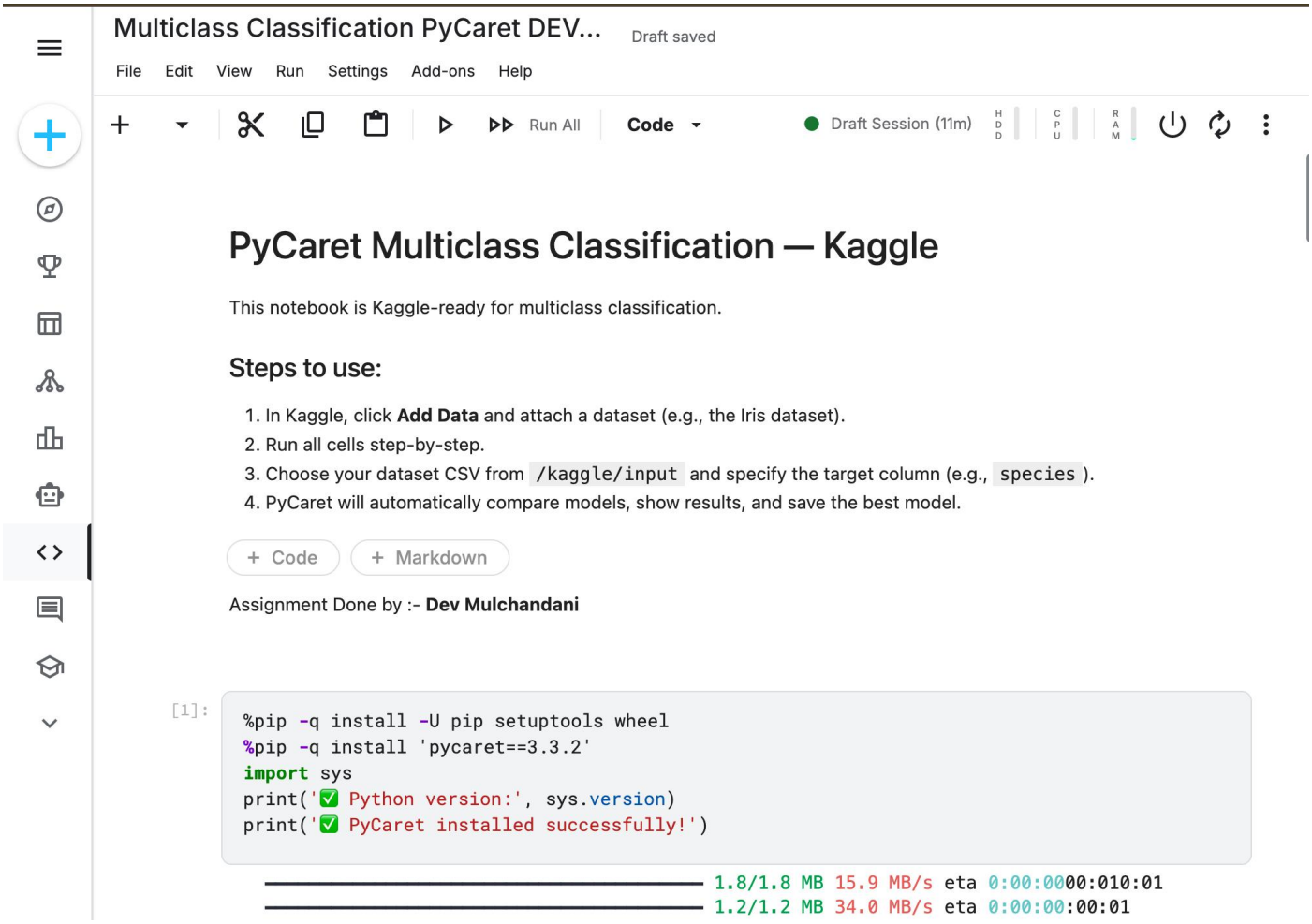
❖ Kaggle Notebook :- Link

❖ Submitted By :- Dev Mulchandani

❖ Overview :-

In my multiclass classification notebook, I used PyCaret's classification module to predict outcomes across three or more categories. Using the Iris dataset, where the target column was Species, I initialized PyCaret with the setup() function to preprocess and prepare the data automatically. I compared several classification models using compare_models() and selected the one with the best accuracy and F1-score. After evaluating the chosen model with confusion matrix and class-wise metrics, I finalized it and generated predictions on new data. This project showcased how PyCaret simplifies multiclass prediction by automating preprocessing, training, and evaluation in a single, streamlined workflow.

❖ Screenshots :-



[2]:

```
import glob, pandas as pd
csvs = sorted([p for p in glob.glob('/kaggle/input/**/*.csv', recursive=True)])
if not csvs:
    raise SystemExit('❌ No CSVs found under /kaggle/input. Attach a dataset first.')
for i, p in enumerate(csvs):
    print(f'{i}: {p}')
idx = int(input('Enter the index of the CSV to load: '))
DATA_PATH = csvs[idx]
print('✅ Using:', DATA_PATH)
data = pd.read_csv(DATA_PATH)
print('📊 Shape:', data.shape)
display(data.head())
```

```
0: /kaggle/input/iris-train/Iris.csv
Enter the index of the CSV to load:  0
✅ Using: /kaggle/input/iris-train/Iris.csv
📊 Shape: (150, 6)
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

[3]:

```
# --- Force CPU mode & avoid RAPIDS imports (must run BEFORE importing PyCaret) ---
import os
os.environ["CUDA_VISIBLE_DEVICES"] = ""  # hide GPUs so libraries skip GPU paths
# If any RAPIDS packages are present, remove them so PyCaret won't try GPU engines
!pip -q uninstall -y cuml cudf cuml-cu12 cudf-cu12 cupy-cuda12x cupy-cuda11x cupy || true
```

```
WARNING: Skipping cuml as it is not installed.
WARNING: Skipping cudf as it is not installed.
```

[4]:

```
from pycaret.classification import *
TARGET = input('Enter the multiclass target column name: ')
exp = setup(data, target=TARGET, session_id=123, use_gpu=False)
best = compare_models()
evaluate_model(best)
save_model(best, 'best_multiclass_model')
```

```
Enter the multiclass target column name:  Species
```

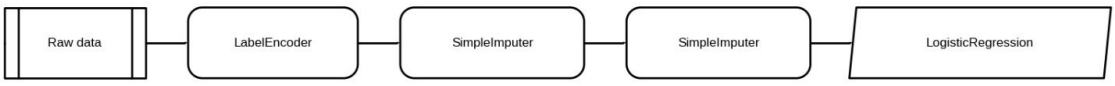
	Description	Value
0	Session id	123
1	Target	Species
2	Target type	Multiclass
3	Target mapping	Iris-setosa: 0, Iris-versicolor: 1, Iris-virginica: 2
4	Original data shape	(150, 6)
5	Transformed data shape	(150, 6)
6	Transformed train set shape	(105, 6)
7	Transformed test set shape	(45, 6)
8	Numeric features	5
9	Preprocess	True
10	Imputation type	simple
11	Numeric imputation	mean
12	Categorical imputation	mode
13	Fold Generator	StratifiedKFold
14	Fold Number	10

16	Use GPU	False
17	Log Experiment	False
18	Experiment Name	clf-default-name
19	USI	c043

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lr	Logistic Regression	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.7410
knn	K Neighbors Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0280
dt	Decision Tree Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0170
rf	Random Forest Classifier	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.1250
ada	Ada Boost Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.0570
gbc	Gradient Boosting Classifier	1.0000	0.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.1080
lightgbm	Light Gradient Boosting Machine	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	7.2620
nb	Naive Bayes	0.9909	1.0000	0.9909	0.9927	0.9908	0.9862	0.9873	0.0170
qda	Quadratic Discriminant Analysis	0.9909	0.0000	0.9909	0.9927	0.9908	0.9862	0.9873	0.0180
lda	Linear Discriminant Analysis	0.9909	0.0000	0.9909	0.9927	0.9908	0.9862	0.9873	0.0160
et	Extra Trees Classifier	0.9909	1.0000	0.9909	0.9927	0.9908	0.9862	0.9873	0.0930
catboost	CatBoost Classifier	0.9909	1.0000	0.9909	0.9927	0.9908	0.9862	0.9873	0.9180
xgboost	Extreme Gradient Boosting	0.9809	0.9888	0.9809	0.9857	0.9806	0.9713	0.9737	0.0420
ridge	Ridge Classifier	0.8600	0.0000	0.8600	0.8859	0.8516	0.7881	0.8049	0.0220
svm	SVM - Linear Kernel	0.5791	0.0000	0.5791	0.5183	0.4895	0.3666	0.4529	0.0210
dummy	Dummy Classifier	0.2864	0.5000	0.2864	0.0822	0.1277	0.0000	0.0000	0.0150

Plot Type:

Pipeline Plot | Hyperparameters | AUC | Confusion Matrix | Threshold
Precision Recall | Prediction Error | Class Report | Feature Selection | Learning Curve
Manifold Learning | Calibration Curve | Validation Curve | Dimensions | Feature Importance
Feature Importanc... | Decision Boundary | Lift Chart | Gain Chart | Decision Tree
KS Statistic Plot



Transformation Pipeline and Model Successfully Saved

[4]:
```
(Pipeline(memory=Memory(location=None),
          steps=[('label_encoding',
                  TransformerWrapperWithInverse(exclude=None, include=None,
                                                transformer=LabelEncoder())),
                 ('numerical_imputer',
                  TransformerWrapper(exclude=None,
                                     include=['Id', 'SepalLengthCm',
                                              'SepalWidthCm', 'PetalLengthCm',
                                              'PetalWidthCm'],
                                     transformer=SimpleImputer(add_indicator=False,
                                                               copy=True,
                                                               fill_value=None,...
                                                               fill_value=None,
                                                               keep_empty_features=False,
                                                               missing_values=nan,
                                                               strategy='most_frequent'))),
                 ('trained_model',
                  LogisticRegression(C=1.0, class_weight=None, dual=False,
```

```python
# 🔮 Predict with the saved multiclass model (robust path picker)
from pycaret.classification import load_model, predict_model
import pandas as pd, glob

# 1) Load your trained model
model = load_model('best_multiclass_model')

# 2) List all CSVs under /kaggle/input and choose by index
csvs = sorted(glob.glob('/kaggle/input/**/*.csv', recursive=True))
if not csvs:
    raise SystemExit("No CSVs found. Click 'Add Data' in Kaggle and attach a dataset, then r
for i, p in enumerate(csvs):
    print(f"{i}: {p}")

idx = int(input("Enter the index of the CSV to use for predictions: "))
test_path = csvs[idx]
print("✅ Using:", test_path)

# 3) Read and predict
test = pd.read_csv(test_path)
preds = predict_model(model, data=test)

# 4) Show a preview
preds.head()
```

```
Transformation Pipeline and Model Successfully Loaded
0: /kaggle/input/iris-train/Iris.csv
Enter the index of the CSV to use for predictions:  0
✅ Using: /kaggle/input/iris-train/Iris.csv
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	Logistic Regression	0.9933	1.0000	0.9933	0.9935	0.9933	0.9900	0.9901

[5]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species	prediction_label	prediction_score
0	1	5.1	3.5	1.4	0.2	Iris-setosa	Iris-setosa	1.0
1	2	4.9	3.0	1.4	0.2	Iris-setosa	Iris-setosa	1.0
2	3	4.7	3.2	1.3	0.2	Iris-setosa	Iris-setosa	1.0
3	4	4.6	3.1	1.5	0.2	Iris-setosa	Iris-setosa	1.0
4	5	5.0	3.6	1.4	0.2	Iris-setosa	Iris-setosa	1.0

[6]:

```python
# Save predictions to CSV
preds[['Id', 'Species', 'prediction_label']].to_csv('predictions.csv', index=False)
print("✅ Predictions saved to predictions.csv")
```

```
✅ Predictions saved to predictions.csv
```