RA2311047010005

INDEX

NAME: Kuberan STD.: AI SEC.: A ROLL NO.: ______ SUB.: Deep learning

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1	24/7	Exploring Deep learning Platform		
2	31/07	Impement a classifier using open source dataset		
3	14/08	study of the classifiers with respect to statistical-parameters		14/8/25
4	14/08	Build a simple feed forward neural network to register uc		

14/08/25
Ex.no:3

3. study of the classifiers with respect to statistical-parameters

Aim

To implement various classifier IRIS dataset and analysis the statistical parameter

Pseudocode

For KNN:.

1. complete the distance x-test, x_i
2. sort all distance in ascending order
3. select first t training points
4. Count frequency of each label
5. Return the label with highest frequency the predicted - class.

For logistic Regression

1. Compute linear combination (z): $z = x_0$
2. Apply sigmoid function: $\hat{y}$ = sigmoid (z) = 1(1+e^{z})
3. Compute loss
4. Compute Gradients
5. Update parameters

$w = w - a^{*} dw$
$b = b - a^{*} db$

For NAYIE BAYES:.

Training phase:.

OBSERVATION

CLASSIFIER	DECISION TREE	SVM	LOGISTIC REGRESSION
Accuracy	84.72%	98.61%	97.50%
Macro Avg precision	85.30%	98.72%	97.67%
Macro Avg Recall	83.98%	98.66%	97.65%
Macro Avg F1-Score	84.31%	98.66%	97.65%
Weighted Avg F1-score	84.72%	98.61%	97.51%

```
1. for each class c in all classes:
   → calculate prior probability
      $P(c) = cont(c) / total+samples$
   → for each feature j:
2. for test point x-test:
      $P(A/B) = \frac{P(B/A)\,P(A)}{P(B)}$
```

OBSERVATION

① → KNN
Accuracy: 160%

② → Logistic Regression
Accuracy: 100%

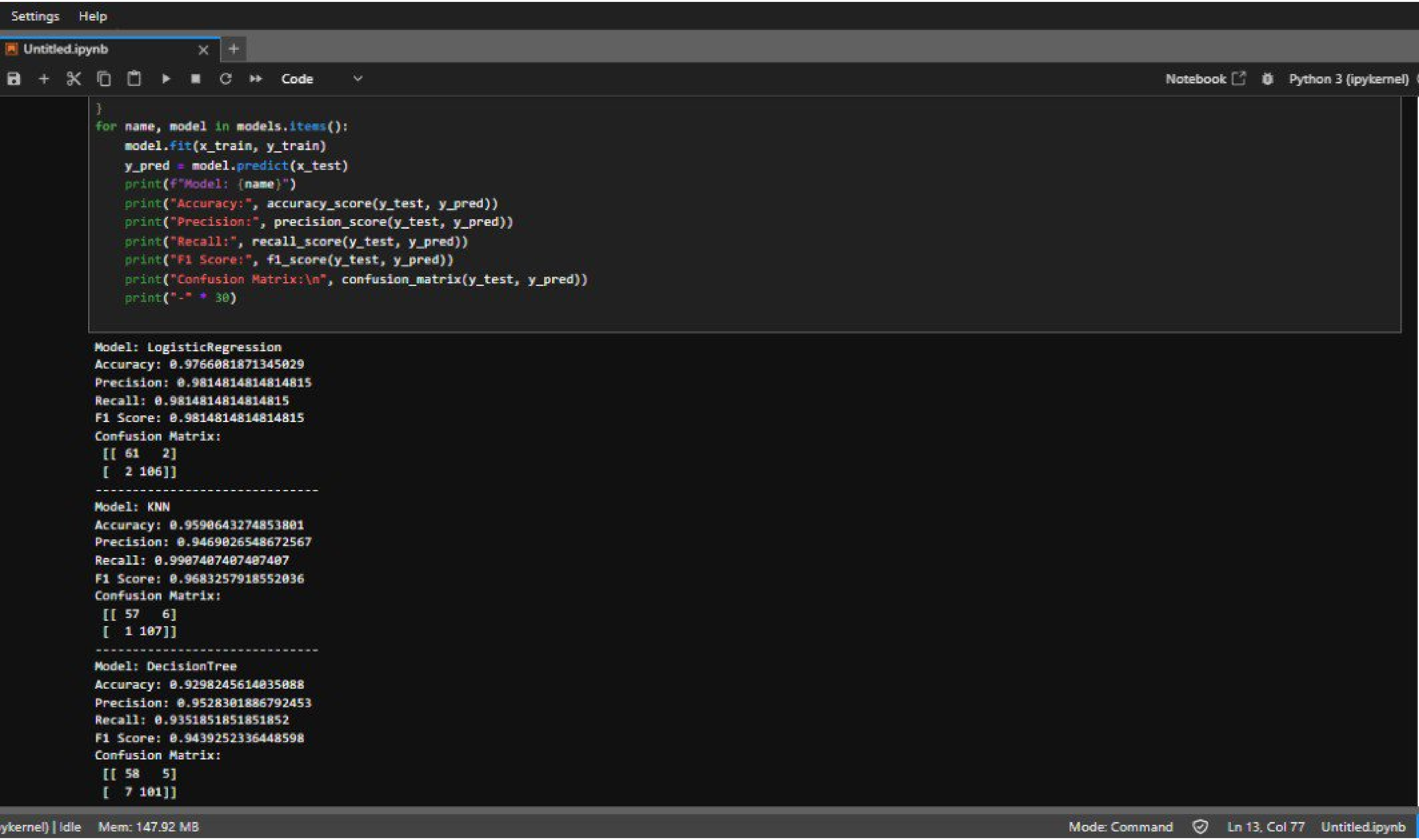
③ → Naive Bayes
Accuracy: 100%

JUSTIFICATION

→ Clear data
→ small samples
→ well separated features
→ Balanced classes

Result

Implemented difference classifier same data-set and analysed Accuracy rat



```python
}
for name, model in models.items():
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    print(f"Model: {name}")
    print("Accuracy:", accuracy_score(y_test, y_pred))
    print("Precision:", precision_score(y_test, y_pred))
    print("Recall:", recall_score(y_test, y_pred))
    print("F1 Score:", f1_score(y_test, y_pred))
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("-" * 30)
```

```
Model: LogisticRegression
Accuracy: 0.9766081871345029
Precision: 0.9814814814814815
Recall: 0.9814814814814815
F1 Score: 0.9814814814814815
Confusion Matrix:
 [[ 61   2]
 [  2 106]]
------------------------------
Model: KNN
Accuracy: 0.9590643274853801
Precision: 0.9469026548672567
Recall: 0.9907407407407407
F1 Score: 0.9683257918552036
Confusion Matrix:
 [[ 57   6]
 [  1 107]]
------------------------------
Model: DecisionTree
Accuracy: 0.9298245614035088
Precision: 0.9528301886792453
Recall: 0.9351851851851852
F1 Score: 0.9439252336448598
Confusion Matrix:
 [[ 58   5]
 [  7 101]]
```