M. Jaya Sai
192425097

① **Aim:-** To compare the performance of Decision tree, and logistic regression, and KNN classifiers on the iris dataset Based on accuracy and execution time.

Algorithm:-

* Load the iris dataset containing sepal and petal measurment
* Split the dataset into training and testing sets
* Train Decison tree, logistic regression, and KNN models using training data.
* Predict the species using test data.
* compare models Based on accuracy and execution time

Program:-

```
from sklearn.datasets import load_risks
from sklearn.model_selection import train_test_split
from sklearn.tree import decisiontree classifier
from sklearn.linear_model import logistic regression
from sklearn.metrices import accuracy_score.
import time.

iris = load_iris()
x, y = iris.data, iris.target.

x_train, x_test, y_train, y_test = train_test split(x, y, test_site)

models = {

    "Decision tree": decisiontree classifier (),
```

```
}
  for name, model in models.items():
      start = time.time()
  model.fit(x_train, y_train)
  y_pred = model.predict(x_test)
  end = time.time()
  print(name, "Accuracy :", accuracy_score(y_test, y_pred)
```

Input:-

iris dataset features; Sepal length, sepal width, Petal length, Petal width.

output:-

Accuracy and execution time for each classifier.

Result:-

All three classifiers successfully classified iris flower, with KNN and logistic regression showing high accuracy.

given input

[5.1, 3.5, 1.4, 0.2]

output

Decision tree accuracy : 0.96 time : 0.002 sec

logistic regression accuracy : 0.97 time : 0.04 sec

KNN accuracy : 0.98 time : 0.003 sec.

② **Aim:-** To implement the candidate elimination algorithm to find all hypothesis consistent with training data.

Algorithm:-

* Initialize the most specific hypothesis S and most general hypothesis G.
* for each positive example, generalize S minimally.
* for each negative example, specialize G minimally.
* remove inconsistent hypotheses from S and G
* output the final version space.

Program:-

```
import pandas as pd
data = pd.read_csv("data.csv")
concepts = data.iloc[:,:-1].values
target = data.iloc[:,-1].values
S = concepts[0].copy()
G = [["?" for _ in S]]

for i, example in enumerate(concepts):
if target[i] == "yes":
    for j in range(lens(S)):
    if example[j] != S[j]:
        S[j] = "?"
```

```
else:
    for j in range(len(s)):
        if example[j] != s[j]:
print("specific hypothesis:",s).
print("General hypothesis:",g).
```

Input:-

citations	size	inlibrary	price	editions	Buy
some	small	NO	affordable	few	NO
many	Big	NO	expensive	many	yes
many	Medium	NO	expensive	few	yes
many	small	NO	affordable	many	yes

Output:-

specific hypothesis: ['many', '?', 'No', '?', '?']

general hypothesis: [['many', '?', 'No', '?', '?']]

Result:-

candidate elimination succesfully produced hypothesis consistent with all training examples.

(3).- **Aim:-** To implement polynomial regression and then evaluate its performance.

Algorithm:-

* load the dataset containing input and output values
* convert input features into polynomial features.
* Train a linear regression model on transformed data
* Predict output values for test data.
* Evaluate the model performance.

Program:-

```
import numpy as np
from sklearn.preprocessing import polynomial_features
from sklearn.linear_model import linear_regression.
x = np.array(1,2,3,4,5).reshape(-1,1)
y = np.array(1,4,9,16,25)

poly = Polynomial_features (degree = 2)
x_poly = poly.fit_transform(x)

model = linear_regression()
model.fit(x_poly, y)

Prediction = model.predict(x_poly)
Print("predicted values", Prediction)
```

Input:-

x = [1, 2, 3, 4, 5]

y = [1, 4, 9, 16, 25]

degree = 2

output:-

Predicted values = [1, 4, 9, 16, 25]

result:-

Polynomial regression accurately modeled the nonlinear relationship in the data.

---

(4):- Aim:- To implement the KNN algorithm for the classification using Python.

Algorithm :-

* Load the dataset and select the value of K
* calculate the distance between test and training points.
* identify K nearest neighbours.
* Assign the most frequent class among the neighbours.
* Display the predicated class.

Program:-

```
from sklearn.datasets import load_iris
from sklearn.neighbours import Kneighbours classifier
iris = load_iris()
x, y = iris.data, iris.target
knn = Kneighbours classifier (n_neighbours = 3)
knn.fit (x, y)
sample = [5.1, 3.5, 1.4, 0.2]]
print("predicated class:", knn.predict (sample)].
```

Input:-

Training data : iris dataset

test sample : [5.1, 3.5, 1.4, 0.2]

K = 3

output:-

Predicated class : Setosa

result:-

KNN successfully classified the input sample based on nearest neighbours.