Date: / /
Page

5/4/24

LAB-1

1. Write a python program to import and export data using pandas library functions.

```
//Importing data.
import pandas as pd
df = pd.read_csv(" ")
df.head(5)

//Exporting data.
url ="https://archive.ics.uci.edu/ml/machine-
      learning_databases/iris/iris.data"

col_names = ["sepallength", "sepalwidth",
             "petallength", "petalwidth",
             "class"]
iris_data = pd.read_csv(url, names= col_names)
iris_data.head(5)
```

2. Demonstrate various data pre-processing techniques for a given dataset.

Algorithm:
- Import the dataset using read_csv
- Identify and handle missing values

solution to handle null values:
① use dropna is drop columns having high number of null values.
② use fillna to replace a null value with specific value.
- Encoding categorical data using pd.get_dum[illegible] which converts categorical data into dummy variables.

Date __/__/__
Page ____

12/4/24

LAB-2

Use an appropriate dataset for building the decision tree (ID3) and apply knowledge to classify a new sample.

ID3 (Examples, Target_attributes)
If all examples are in same class, return a leaf node with that class label.
If the list of attributes is empty, return a leaf node with the most common class.

Choose the best attribute A to split on using entropy and information gain
Entropy of the entire dataset $S(+,-) = -P_{\oplus} \log_2 P_{\oplus} - P_{\ominus} \log_2 P_{\ominus}$
Information gain = entropy (parent) − $\sum_{i=1}^{n}$ weighted average * entropy (child);

for each possible value v of A:
  add a new branch below the decision node for value v.
  let examples_v be the subset of examples with value v for attribute A.
  If Examples_v is empty:
    add a leaf node with the most common class label in examples to this branch.
  else:
  recursively call ID3 (examples_v, Target attributes) and add the returned subtree to this branch.
Return the decision tree.

Output:
① Highest information gain = 0.246 = outlook.
② Highest information gain = 0.971 = rainy.
Best attribute is windy.

3/5/2024

Date
Page

3/5/24

LAB-3

1. Build KNN classification model for given dataset.

Algorithm:

1) Define the value of K and a distance metric

2) For the given point, calculate the distance between the given point and every other point in the dataset.

3) Choose K, closest points.

4) The class/value of the given point is the majority of that of K points.

If euclidian distance is used as distance metric:

then $d = \sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$

Output:

Our model accuracy = 0.9667.

model.predict([[7, 7, 2.6, 6.9, 2, 3]])

array(['Iris-virginica'], dtype='')

2. Build linear regression model for a given set.

Algorithm:

function linear regression (X, Y, learning rate).

Initialize random values for slope (m) & intercept (b)

for i = 1 to num-iterations:

step 2: compute predictions.

predictions = m * X + b.

3/5/2024

# step3: compute error.
errors = predictions - y.
# step 4: compute loss function
loss = mean_squared_error(errors)
# step 5: Gradient Descent.
return w, b.

function mean squared error(errors)
mse = sum(errors²)/len(errors)
return mse.

3. Implement logistic Regression for a given set.

Algorithm:

function logistic_regression(X, y, learning rate)
- Initialize value for weights (w) and bias (b)

for i=1 to num_iteration:
logits = X * w + b.
predictions = sigmoid(logits).
loss = compute_loss(y, predictions)
update weights and bias using gradient
return w, b.

function sigmoid(X)
return 1/(1+ exp(-X)).

Output:
Accuracy: 78.45.

3/5/2024

24/5/24

LAB-4

1. SVM (Support vector machine)

① Define the kernel function..

Eg: $K(x_1, x_1) = x_1 . x_2$

② Solve quadratic programming problem to find the α.

③ Compute the weight and bias

④ Identify the support vectors.

⑤ Make prediction.

Output

```
→ model = SVM()
  model.fit (X_train, y_train)
  predictions = model.predict (X_test)
  accuracy (y_test, predictions)
0.98230088
```

2. K-means clustering algorithm

① Select the number K to decide the number of clusters.

② Select random K points on centroids.

③ Assign each point to the closest centroid. from predefined K cluster.

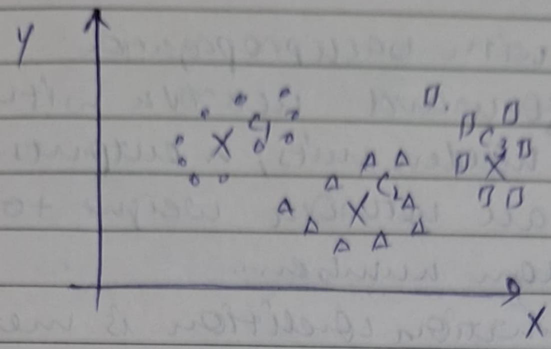
④ Calculate the variance and place new centroid of each cluster.

⑤ Repeat ③ step, reassign the centroid

⑥ If any assignment occurs go to step ④ else goto finish.

⑦ Model is ready.

Output



3. PCA (Principal Component Analysis.
① Calculate mean.
② Calculation of covariance matrix
③ Eigen values of covariance matrix.
④ Computation of eigen vector/unit eigen vectors.
⑤ Computation of first principal component
⑥ Geometrical meaning of first principal component.

Output
→ pca.explained_variance_ratio_

array([0.98377428, 0.01620798])

31/5/24

Lab 5

1. Build ANN with backpropagation.

(1) Create feed forward network with n inputs, n hidden units, n outputs.

(2) Initialize all network weight to small random numbers.

(3) Until termination condition is met, DO

- For each $(\vec{x}, \vec{t})$ in training example
- Propagate input forward.
- Propagate error backward.
- For each hidden unit, calculate error.
- Update weights.

Output

Testing accuracy 8/9 = 0.88

2. Random forest

(1) Import libs, load dataset

(2) train, test split

(3) Train the data.

(4) Initialize random forest regressor

(5) Train & make predictions

(6) Evaluate using MSE

Output

Accuracy = 0.93

3. Adab

31/5/2024

Date __/__/__
Page ______

3. Adaboost Algorithm.

1. Import libs, load dataset
2. Initialize Adaboost model.
3. Train the model.
4. Make predictions
5. Evaluate model on metrics like mean absolute error.

Output

Accuracy = 0.944

31/3/2024