JOY
PAGE NO.
DATE / /

Lab-01

Write a python program to import & export data using pandas library functions

→ Pandas read csv()

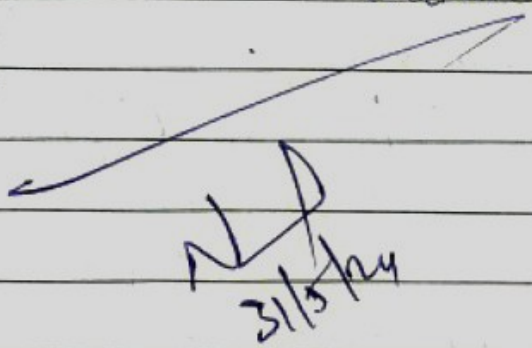
```
import pandas as pd.
airbnb_data = pd.read_csv("austin Housing Data.csv")
airbnb_data.head()
```

Exporting dataframe to a csv file.

```
iris_data.to_csv("cleaned_iris_data.csv")
```

importing dataset using URL

```
Url = "https://archive.ics.uci.ml/machine-learning
-dataset/iris.data"
```



31/5/24

JOY
PAGE NO.
DATE / /

Lab 02

Demonstrate various data preprocessing techniques

Algorithm

1) Downloading the dataset.
2) Reading csv file
   pd.read_csv(data_path)
3) Observing dataset using head(), info() & describe()
4) Visualization of dataset using matplotlib.
5) Create test set & train_set_split(housing, test_size=0.2, random_state=42)
6) Visualise the data to gain insights.
7) find the correlation bet the categories
8) Data cleaning by dropping N/A values
   housing.dropna()
9) Imputation of missing values
10) Encode categorical values with nos
11) Scaling data using standardization or min-max strategy
12) Training the linear regression model
13) Calculating root mean square error
14) Training decision tree
15) Cross validation using mean & standard deviation
16) Fitting test data & calculating accuracy.

12-4-24

PAGE NO.
DATE / /

Q. Implement linear regression algorithm using appropriate dataset.

Algorithm

1. Import necessary libraries.
2. Import dataset
3. Visualization of dataset using diff plots like heatmap, distribution plot, scatterplot etc.
4. preprocess the data, convert or encode categorical data
5. Split the dataset into training set & test set.

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split (x, y, test_size = 0.3, random_state = 23)

31/5/24

6. Build model

from sklearn.linear_model import LinearRegression

lin_reg = LinearRegression()

7. Fit the dataset to model & train it

lin_reg.fit (X_train, y_train)

8. Calculate the accuracy using mean square error

PAGE NO
DATE / /

Q. Implement multilinear regression.

1) Import necessary libraries like LinearRegression
2) Import dataset
3) Visualize the data set using matplotlib.pyplot
4) Encode categorical data
ct = ColumnTransformer(transformers = [('encode', OneHotEncoder[3 3)], remainder = 'passthrough')
5) Split dataset to training & test dataset
6) We can see multiple independent variables
7) Create regressor model.
regressor = LinearRegression()
8) Fit the train set.
9) Test the model using test set.
10) Compare the actual value & predicted value

31/5/24

JOY
PAGE NO.
DATE / /

Q) Use appropriate dataset for budding the decision tree ID3 & apply the knowledge to classify new sample

1) Import the libraries like Decision Tree Classifier.
2) Import diabetes dataset.
3) Define calculate_entropy function to calculate entropy for each value
v_count = no.of values.
proportion = v_count / total_rows
entropy = proportion * math.log2 (proportion)

4) Define information_gain function
information_gain = entropy_outcome → weighted_entropy

5) Access the feature with highest information gain.

6) Create a decision tree using DecisionTree Classifier function

7) Plot the tree

8) Define ID3 function & pass new sample as parameter & obtain classification

PAGE NO.
DATE / /

Q. Build logistic regression model for given dataset.

1) Import all required libraries.
2) Import the given dataset
3) Preprocess the dataset to standard scale
4) Split the dataset into test & train
5) Code the logistic Regression model

```
LR = LogisticRegression(C=0.01, solver='liblinear').
     fit(x_train, y_train)
```

6) Predict the test set using the model

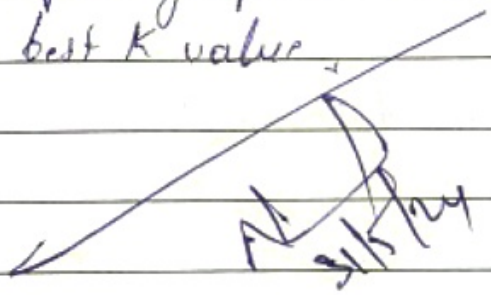
```
yhat = LR.predict(x_test)
yhat_prob = LR.predict_proba(x_test)
```

7) Calculate the performance or accuracy of the models.

PAGE NO.
DATE / /

Q. Build KNN classification model for a given dataset

1) Import the necessary libraries
2) Import diabetes dataset.
3) Preprocess the dataset.
4) Split the dataset as test & train set
5) Build the KNN model.
   KNN = KNeighbors Classifier (n_neighbours = k)
6) Fit the model & test the accuracy.
7) We can plot graph of K v/s accuracy to choose the best K value.



31/5/24

31-5-24

7) Build svm model for a given dataset

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()

print("Features:", cancer.feature_names)
print("Labels:", cancer.target_names)

cancer.data.shape

print(cancer.data[0:5])

print(cancer.target)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, test_size=0.3,
    random_state=109)

from sklearn import svm
clf = svm.SVC(kernel='linear')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

from sklearn import metrics
print("Accuracy:", metrics.accuracy_score
                    (y_test, y_pred))

print("Precision", metrics.precision_score
                    (y_test, y_pred))
print("Recall", metrics.recall_score(y_test, y_pred))
```

JOY
PAGE NO.
DATE / /

8) Random Forest algorithm

→ Data preprocessing step
→ fitting the random forest algorithm to training set
→ predicting the test result
→ test accuracy of the result (Creation of confusion matrix)
→ Visualizing the test set result

Step 1:- Select random k data points from the training set
Step 2: Build the decission trees associated with the selected data points
Step 3: Choose the no. N for decision trees that you want to build
Step 4: Repeat step 1 & 2
Step 5: For new datapoints, find the predictions of each decision tree & assign new datapoints to category that wins the majority votes

3/6/24

a) Implement boosting ensemble method on given dataset

1) → Initialize weights :- assign equal weights to all training samples

2) → For each base learner (weak learner):
train a weak learner on training data with current weights

3) → Update sample weights :- Increase weights of the incorrectly classified samples so that they are more likely to be selected in next iteration

→ Repeat steps 2 & 3 for a fixed no. of iterations or until a stopping criterion is met

→ Aggregate the predictions

→ output the ensemble model

DATE / /

10) Build K-means algorithm to cluster a set of data stored in a CSV file

function kMeans(X, K, max_iterations):

1) Initialize centroids randomly
2) Repeat until convergence or max_iterations:
   a. assign each data point to nearest centroid
   b. Update centroids based on the mean of data points in each cluster
   c. check convergence
3) Return the final centroids

11) Dimensionality reduction using PCA

Standardize data :- compute mean μ & standard deviation σ for each feature.

2) Compute the covariance matrix

$$\Sigma = \frac{1}{m}(X-\mu)^T(X-\mu)$$

3) Compute eigenvectors & eigenvalues:
4) Select the top K eigenvectors:
   Select the top K eigenvectors corresponding to largest eigenvalues to form projection matrix W
5) Project the data onto the new feature space:

$$X_{pca} = X \cdot W$$

6) output :- return the transformed dataset X_{pca}.