Practical Machine Learning Lab

Lab Manual with Student Lab Record

Developed By

Dr. K. Rajkumar

P. Bhuvaneswari

Roll No	
Name	
Class	**I MSc Data Science**

Department of Data Science

Bishop Heber College (Autonomous)

Tiruchirappalli 620017, INDIA

March 2021

Copyright © Bishop Heber College

Department of Data Science
Bishop Heber College (Autonomous)
Tiruchirappalli 620017

BONAFIDE CERTIFICATE

Name:_____

Reg. No:_____ **Class:**_____

Course Title _____

Certified that this is the bonafide record of work done by me during **Odd / Even** Semester of **2020 – 2021** and submitted for the Practical Examination on _____

Staff In-Charge **Head of the Department**

Examiners

1._____

2._____

Grade Sheet

Roll No		**Name**	
Year		**Semester**	
		Instructor Name	

Lab	**Date**	**Activity**	**Grade**
1		WarmUp: Familiarity with Data and Visualization	
2		Pizza Liking Prediction using kNN	
3		Fuel Amount Prediction using Linear Regression	
4		House Price Prediction using LR with Regularization	
5		Diabetes Classification using Logistic Regression	
6		Predictive Analytics for Hospitals	
7		Loan Approval Classification using SVM	
8		Animal Classification using Decision Trees	
9		Employee Hopping Prediction using Random Forests	
10		Patients Physical Activities Prediction using Boosting	
11		Shopping Mall Customer Segmentation using Clustering	

Preface

This laboratory manual is written to accompany the lab course titled, *Practical Machine Learning Lab*. The aim of this laboratory manual is to help students to enhance the understanding of concepts presented in class and to solve problems outlined in the syllabus of a lab course.

The lab exercises have been grouped into weekly activities for a semester. The weekly lab sheets include: input, output, source code and extra credit activities.

Students using these lab sheets should note the following:

1. Fill out your roll no and name in all required places.
2. Read carefully all details of an exercise of a week.
3. Understand the source code which may be a complete code or just a code snippet.
4. The required updates would have been included as comments inside the source code. You need to update them so that your code is ready for execution.
5. Once your code is executable, run your code with the test case inputs and get the results. Verify your obtained output against the expected output.
6. Now, carefully read all extra credit activities, revise your code accordingly, rerun your source code and obtain new outputs.
7. Upon solving all exercises including extra credit activities, approach your lab instructor, demonstrate your experiments and get your grades for the lab.

Final note, attend your weekly lab session with your lab manual without fail. Also, it is your responsibility to keep your lab manual safe as it records the grade you received every week. Comments about the laboratory exercises presented in this Lab manual are welcomed and encouraged. We hope that you will overlook any misspellings, omissions, errors and inconsistencies and report such issues to us. Happy coding!

Dr. K. Rajkumar

Lab1: Warm Up – Familiarity with Data types and Visualization

Objectives

In this lab, you will get familiarity with downloading, reading, printing properties and visualizing datasets. Also, you will work on various Notebooks such as Google Colab and Azure notebooks.

Learning Outcomes

After completing this lab, you will be able to

- Understand various data formats
- Understand various file formats
- Visualize various kinds of data such as text, images, video and audios
- Get familiarity with Google Colab and Azure notebooks

Step 1: Download the dataset files that belong to the following data formats from internet. The files may belong to any dataset available online.

Step 2: Read these files inside the python code. Some of the file formats cannot be read using default python packages. In this case, explore the python packages suitable for reading the files.

Step 3: Print the properties of the data files such as size, shape, dimensions, etc.

Step 4: Visualize each of these data files using graphs, diagrams, etc.

- Table data visualization: line graph, bar graph, histogram chart, pie chart, scatter plot
- Image visualization: image plot, 3d plot
- Video visualization: video player
- Audio visualization: audio player, spectrogram
- Text visualization: Word cloud, bubble cloud (some more in http://vallandingham.me/textvis-talk/)

1. Tabular, Spreadsheet and Interchange Data Formats

- "Table" — generic tabular data (.dat), "CSV" — comma-separated values (.csv), "TSV" — tab-separated values (.tsv), “ARFF” - Attribute-Relation File Format (.arff) – Read and visualize the data
- "XLS" — Excel spreadsheet (.xls), "XLSX" — Excel 2007 format (.xlsx), "ODS" — OpenDocument spreadsheet (.ods), "SXC" — OpenOffice 1.0 spreadsheet file (.sxc), "DIF" — VisiCalc data interchange format (.dif) – Read and visualize the data
- "JSON" — JavaScript Object Notation (.json), "UBJSON" — Universal Binary JSON (.ubj), “HTML” – Hypertext Markup Language (.html), “XML” - eXtensible Markup Language (.xml) - Read and Parse the data

2. Data File Formats

- PKL – Pickle format, HDF5, Zip, SQL, MAT, NPY, NPZ – Read and display the data

3. Image Data Formats

- JPG, PNG, BMP, TIFF – Read and display the image
- 3D medical Images: DICOM, MHA – Read and display the image

4. Video Data Formats

- MP4, AVI, MPEG – Read and play the video

5. Audio Data Formats

- MP3, MIDI, WAV – Read and play the audio

6. Text Data Formats

- TXT, PDF, DOC – Read and parse the data

Familiarity: Get familiarity with **Google Colab Notebook** and Microsoft **Azure Notebooks.**

NOTES

Lab2. Pizza Liking Prediction using kNN

Objectives
In this lab, you will build a k-Nearest-Neighbour model to predict whether a person will like pizza or not based on his age and weight.

Learning Outcomes
After completing this lab, you will

- Identify features (aka variables – dependent and independent) and create dataset
- Import dataset and understand properties such as size, data types and others
- Use sklearn and instantiate kNN model
- Perform fit and predict operations
- Compute accuracy
- Select the best value for K

Business Use Case
You are a Data Scientist. A local Pizza Restaurant in your city has approached you with the details of its customers such as age and weight and whether they liked their Pizza or not. The restaurant wanted you to help them to develop a system that will predict whether a customer will like their Pizzas given their age and weight. The restaurant has given you the following dataset for assessing your predictive analytics skills.

Step1. [Prepare your dataset]
Using MS Excel, create the following CSV file and save it as, **"pizza.csv"**

age	weight	likePizza
50	65	0
20	55	1
15	40	1
70	65	0
30	70	1
75	60	0

Create another CSV, as below, but with age and weight columns only and save it as, **"pizza_test.csv".**

age	weight	likePizza
48	68	1
35	45	1
15	40	0
55	65	0

Step2. [Import dataset]. Using Pandas, import **"pizza.csv"** file and print properties such as head(), shape, columns and info.

Step3. [Visualize Relationships]. Plot relplot between "age" and "weight", with hue as "likePizza"

Step4. [Prepare X matrix and y vector]. Extract "age" and "weight" columns and store into new dataframe **X**. Similarly, extract "likePizza" column and store into **y**.

Step5. [Examine X and y]. Print X, y, type of X and type of y.

Step6. [Model building]. Create **KNeighborsClassifier(n_neighbors=2)** from **sklearn** and perform **fit** on X and y.

Step7. [Model testing]. Using your KNN model, predict if a person will like Pizza or not.

- Will a person who is 25 years with weight 50 kgs like Pizza or not? – The answer should be 1 (ie., YES)
- Will a person who is 60 years with weight 60 kgs like Pizza or not? – The answer should be 0 (ie., NO)

Step.8 [Change n_neighbors = 3]. Now, create new model, perform fit and predict steps. Check results for the above 2 queries. Are they same?

Step9. [Predict on entire dataset]. Now, perform prediction on entire X matrix and store result as y_pred.

Step10. [Accuracy function]. Create a ***function accuracy()*** and returns accuracy. The accuracy function can be defined as follows:

```
def accuracy(actual, pred):
    return sum(actual == pred) / float(actual.shape[0])
```

Step11. [Find accuracy]. Call accuracy() with **y and y_pred** as parameters and print accuracy score. Are you getting score as **1.0** ?

KNN predicted all samples in X correctly. Because, you created KNN model with those values of X and y. But, it may not be the same if you use different samples for testing.

Step12. [Prediction on Test Set]

Using Pandas, import **"pizza_test.csv"** file and print properties such as head(), shape, columns and info.

Using KNN model with n_neighbors=2, that you created previously, perform prediction on X values from pizza_test dataframe. Call accuracy function and print accuracy score. Are you getting a score of **0.5**? That is, our model has predicted 2 samples correctly and two wrongly, out of 4 samples in the test set.

Step13. [Find best value for k]. If you want to improve the accuracy of your model, then you should use the best value k for the nearest neighbors.

Now, let us rerun our KNN for various values of k (k = 1 to 3) and find accuracy scores. Then, we will be able to select the best k for which accuracy score is the highest.

```
scores = list()

for k in range(1, 4):
   create knn model
   perform fit on X and y
   predict test set X
   find accuracy on y values of test set and predicted values
   store k and accuracy as a tuple in scores list
```

When you print scores list, you will get the following output.

```
[(1, 0.5), (2, 0.5), (3, 0.5)]
```

Step14. [accuracy_score function]. Call accuracy_score() function with y_test and y_pred values. You can import as "*from sklearn.metrics import accuracy_score*".

CONCLUSION

Our accuracy values remain same for all values of k. Because, our dataset is so small. When working on larger datasets, your will find accuracy improving on various values of k.

NOTES

NOTES

Lab3. Fuel Amount Prediction using Linear Regression

Objectives

In this lab, you will build a Linear Regression model to predict the amount to be spent on fuel (ie., diesel) for your car for a road trip.

Learning Outcomes

After completing this lab, you will

- Acquire data and prepare dataset
- Split dataset for training and testing
- Build LR model, perform training, testing and compute MSE and R2 error values
- Apply normalization methods for feature scaling
- Compare performance with KNN Regressor and SGD Regressor

Business Use Case

Say you're planning a road trip to **Goa** with two of your best friends. You start off in **Trichy** and you know it's going to be a ~20h drive. While your friends are in charge of the party operations you're in charge of the all the logistics involved. You have to plan every detail: the schedule, when to stop and where, make sure you get there on time.

So, what's the first thing you do?. You get yourself a blank sheet of paper and start planning!. First item on your checklist? Budget! It's a 20h — approximately 800 kilometers — fun ride, so a total of 20h on the road. The follow-up question: **How much money should I allocate for diesel?**

This is a very important question. You don't want to stop in the middle of the highway and possibly walk a few miles just because you ran out of diesel ! How much money should you allocate for diesel ?.

You approach this problem with a science-oriented mindset, thinking that there must be a way to estimate the amount of money needed, based on the distance you're travelling.

First, you look at some data. You've been laboriously tracking your car's efficiency for the last year — because who doesn't! — so somewhere in your computer, you have stored in an Excel sheet (***fuel_data.csv***) , as shown below.

drivenKM	fuelAmount
390	3600
403	3705
396.5	3471
383.5	3250.5
321.1	3263.7
391.3	3445.2
386.1	3679
371.8	3744.5
404.3	3809
392.2	3905
386.43	3874
395.2	3910
381	4020.7
372	3622
397	3450.5
407	4179
372.4	3454.2

375.6	3883.8
399	4235.9

Step1. [Prepare your dataset]. Create ***fuel_data.csv*** file as shown above.

Step2. [Import dataset]. Using Pandas, import **"*fuel_data.csv*"** file and print properties such as head(), shape, columns, type and info.

Step3. [Preprocessing]. Check for missing values (Use isnull() method)

Step4. [Visualize Relationships]. Plot ***relplot*** between "**drivenKM**" and "**fuelAmount**".

Step5. [Prepare X matrix and y vector]. Extract "**drivenKM**" column and store into new dataframe **X**. Similarly, extract "**fuelAmount**" and store into **y**.

Step6. [Examine X and y]. Print X, y, type of X and type of y.

Step7. [Split dataset]. Split dataset into 4 parts using ***train_test_split()*** method, such as *X_train, X_test, y_train* and *y_test*. Use **20%** for test size. Later you can play around with this test size. Print the shape of all 4 parts.

Part-I. Linear Regression Baseline Model

Step8. [Build Model]. Create Linear Regression model and train with ***fit()*** using X_train and y_train values.

Step9. [Predict price for 800 KM]. If I need to travel 800 KM, how much do I need to spend on Diesel?. Are you getting this ouput, **array([6905.64571567]). ?**

Step10. [Predict on entire dataset]. Now, perform prediction using entire **X_test** and store result as **y_pred**.

Step11. [Print Mean Squared Error and R2 Error]. Are you getting output **"MSE: 46181.0".** Also, print values of model parameters: ***coef_*** and ***intercept_*** values.

Part-II. Linear Regression with Scaling using StandardScaler

Step12. [Normalize X_train and X_test values]. Use **StandardScaler**, scale X_train using **fit_transform()** method and X_test using **transform()** method.

Step13. [Build LR model]. Create a new LR model, fit on *scaled X_train* and predict on *scaled X_test*.

Step14. [Print Mean Squared Error and R2 Error]. What is the output?. MSE reduced or not?. Why?.

Step15. [Plot scatter plot]. Display Scatter Plot between actual y (aka ground truth) vs predicted y values. That is, between ***y_test* and *y_pred***.

Part-III. Linear Regression with Scaling using MinMaxScaler and Comparison with KNeighborsRegressor and SGDRegressor

Step16. [Repeat with MinmaxScaler]. Repeat scaling using **MinMaxScaler**, LR model creation, fit, predict and error computation steps.

Step17. [Compare KNN Regressor]. Repeat the above steps for **KNeighborsRegressor** model and compare MSE of LR with KNN Regressor.

Step18. [Compare SGD Regressor]. Repeat the above steps for **SGDRegressor** model and compare MSE of LR with SGD Regressor.

Step19. [Select best model]. Tabulate MSE values of LR, KNNR and SGDR and select the model with the lowest MSE.

NOTES

NOTES

Lab4. House Price Prediction using LR with Regularization

Objectives
In this lab, you will learn how to build Regression Models to predict the Sales Price of houses using Ames, Iowa dataset.

Learning Outcomes
After completing this lab, you will be able to

- Understand data and build baseline LR model
- Apply One Hot Encoding to categorical features and rebuild LR model
- Apply scaling using StandardScaler and MinMaxScaler and rebuild LR model
- Compare the performance of LR model with SGD Regressor, RidgeCV and LassoCV
- Compute RMSE for all models.

Business Use Case
Ames house sales price dataset contains past sales price details of 1379 homes from Ames, USA. This dataset has 79 feature columns (independent variables) and the dependent variable, sale price ("**SalePrice**"). There are three different types: integers (int64), floats (float64), and strings (object, categoricals).

In this lab, you will use a cropped dataset where we have removed all object columns, except two object columns – building type ("**BldgType**") and centralized air conditioning ("**CentralAir**"). Therefore, it contains only 38 independent variables.

With the help of the regression model you will build on this reduced dataset, we can predict the sale price of a house given the values for all independent variables.

Step1. [Import dataset]. Using Pandas, import **"Ames_House_Sales_Cropped.csv"** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Predict Sale Price without Categorical features].

- Drop both categorical features – BldgType and CentralAir (USE drop() and pop() methods)
- Prepare X matrix (36 feature columns) and y vector (ie., SalePrice column)
- Split dataset for training and testing as X_train, X_test, y_train, y_test (use **25%** test size).
- Create LinearRegression model, fit on training set and predict on test set
- Compute Mean Squared Error (MSE) on actual values and predicted values (you will get output as 1474827326.0).

Step3. [Create Scatter Plot]. Plot Scatterplot between **y_test** and **y_pred**.

Step4. [Encode Categorical columns]. Using **get_dummies()** method, perform one hot encoding on the two categorical columns, *BldgType* and *CentralAir*. Now, you will get 5 columns for BldgType variable and 2 columns for CentralAir column. So, now you have 43 independent variables and 1 dependent variable.

Step5. [Predict Sale Price with Categorical features]

- Prepare X matrix (43 feature columns) and y vector (ie., SalePrice column)
- Split dataset for training and testing
- Create LinearRegression model, fit on training set and predict on test set
- Compute Mean Squared Error on actual values and predicted values (you will get output as 1461036570.0).

Step6. [Normalize using StandardScaler and Predict Sale Price]

- Using StandardScaler, perform **fit_transform()** on X_train and **transform()** on X_test matrix that you already splitted.

- Create a new Linear Regression model, fit on *scaled X_train* and *y_train* and predict on *scaled X_test*.
- Compute Mean Squared Error (MSE) on actual values and predicted values (you will get output as 1461036570.0).

Step7. [Normalize using MinMaxScaler and Predict Sale Price]

- Repeat Step6 using MinMaxScaler
- Mean Squared Error will be: 1461036570.0

Step8. [Predict using SGD Regressor]

- Use scaled X_train and X_test using StandardScaler that you computed before
- Create SGDRegressor, fit and predict
- Compute MSE on y_test and y_pred this time. You will get output as 1592430104.0.

Step8. [Predict using Ridge Regression]

- Use scaled X_train and X_test using StandardScaler that you computed before
- Create RidgeCV, fit and predict
- Compute MSE on y_test and y_pred this time. You will get output as 1442196000.3367693.

Step8. [Predict using Lasso Regression]

- Use scaled X_train and X_test using StandardScaler that you computed before
- Create LassoCV, fit and predict
- Compute MSE on y_test and y_pred this time. You will get output as 1409368613.5329669.

Step9.[RMSE]. Print Root Mean Squared Error values (use numpy.sqrt() method) as below and compare error values.

RMSE without one hot encoding: 38403.0
RMSE with One hot encoding: 38224.0
RMSE with OHE and Standard Scaling: 38224.0
RMSE with OHE and MinMax Scaling: 38224.0
RMSE of SGDRegressor with OHE and Standard Scaler: 38528.0
RMSE of RidgeCV with OHE and Standard Scaler: 37976.0
RMSE of LassoCV with OHE and Standard Scaler: 37542.0

NOTE

NOTE

Lab5. Diabetes Classification using Logistic Regression

Objectives

In this lab, you will classify using Logistic Regression model, whether a person will become diabetic or not using PIMA diabetes dataset available in UCI. You will also predict a new person who is not in the dataset will become diabetic of not based on his details.

Learning Outcomes

After completing this lab, you will be able to

- Understand data and build baseline LogisticRegression model
- Create Heatmap with confusion matrix values
- Apply scaling using StandardScaler and MinMaxScaler and rebuild LoR model
- Print classification metrics scores and plot ROC curve
- Compare the performance of LoR model with LogisticRegressionCV with L1 and L2 regularization

Business Use Case

You are a data scientist. A leading hospital in your city has approached you with the medical details of their patients related to their diabetes information. The hospital has given you a file (diabetes.csv) that contains details of 768 patients and each patient is described with these 9 features. Here, ***Outcome*** is the ***dependent variable*** and ***all other 8 variables are independent variables.***

- Pregnancies: Number of times pregnant
- Glucose: Plasma glucose concentration over 2 hours in an oral glucose tolerance test
- BloodPressure: Diastolic blood pressure (mm Hg)
- SkinThickness: Triceps skin fold thickness (mm)
- Insulin: 2-Hour serum insulin (mu U/ml)
- BMI: Body mass index (weight in kg/(height in m)2)
- DiabetesPedigreeFunction: Diabetes pedigree function (a function which scores likelihood of diabetes based on family history)
- Age: Age (years)
- Outcome: Class variable (0 if non-diabetic, 1 if diabetic)

The hospital wants you to build a model so that the model will assess when a new patient visits them he will become diabetic or not.

Step1. [Understand Data]. Using Pandas, import **"diabetes.csv"** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Build Logistic Regression Model]

- Prepare X matrix (8 feature columns) and y vector (ie., Outcome column)
- Split dataset with stratified shuffle split for training and testing as X_train, X_test, y_train, y_test (use **25%** test size).
- Create LogisticRegression model, fit on training set and predict on test set

Step3. [Predict on a new sample]

- Will this person become diabetic?. His details are given below.
- new_person = [[6, 200, 90, 10, 25, 23.3, 0.672, 42]]

Step3. [Compute Classification Metrics]

- Compute and print Accuracy, Precision, Recall and AUC scores

Step4. [Understand Correlation]

- Create confusion matrix between y_test and y_pred and plot confusion matrix values in a Heatmap. Explain the meaning of the 4 numbers you get.

Step5. [Normalization using MinmaxScaler and rebuild LoR]

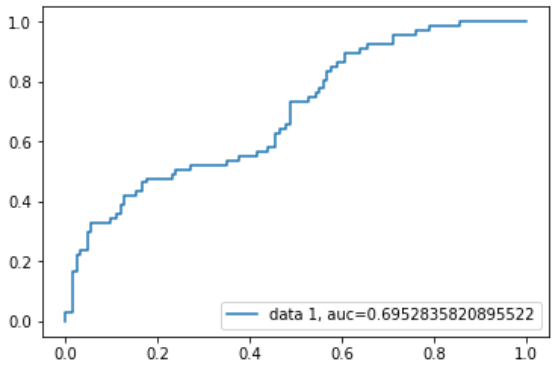
- Now, normalize your X_train and X_test values using MinmaxScaler
- Create a new LogisticRegression model, fit on normalized training set and predict on the normalized test set
- Compute and print Accuracy, Precision, Recall and AUC scores

Step6. [Normalization using StandardScaler and rebuild LoR]

- Repeat Step5 with StandardScaler
- Among the 3 models, which model gives better classification scores?

Step7. [Plot ROC curve]

- Plot ROC curve as shown below. You can use the MinmaxScaler scaled values of X_test for computing predict_proba() score.

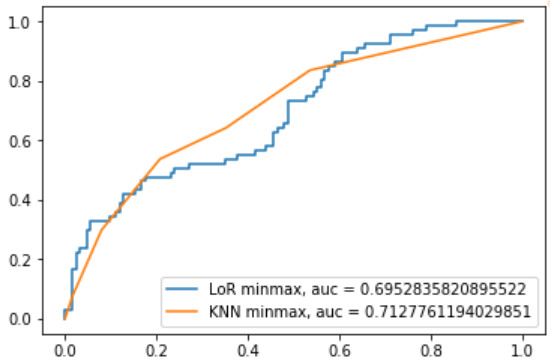


Step8. [Comparison with KNN classifier].

Create a KNN classifier with default values, fit on the scaled X using MinmaxScaler, predict and print classification metric scores.

Step9. [Update ROC curve]

- Update your ROC curve, this time, with one more curve of KNN classifier, as shown below.

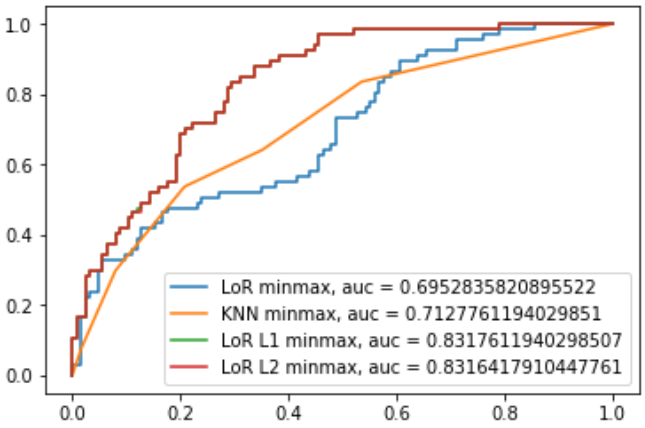


Step10. [Regularization]

- In order to reduce overfitting of your data, you will use *LogisticRegressionCV* model with L1 and L2 regularization parameters. Create both models using the following statements
  - `model1 = LogisticRegressionCV(Cs=10, cv=4, penalty='l1', solver='liblinear')`
  - `model2 = LogisticRegressionCV(Cs=10, cv=4, penalty='l2')`
- Perform fit using MinmaxScaler scaled values and predict

Step11. [Update ROC curve]

- Update your ROC curve, this time, with two more curves, as shown below



Out of these 4 models, which model performs the best?. How?. Why?.

NOTES

NOTES

Lab6. Predictive Analytics for Hospitals

Objectives

In this lab, you will continue to work on Diabetes data from Lab4 and apply Predictive Analytics principles to recommend the course of action to be taken by the hospital.

Learning Outcomes

After completing this lab, you will be able to

- Perform prediction on single and multiple independent variables for the target
- Apply Forward Selection procedure to find the best features based on AUC scores
- Draw a line plot of AUC scores for variables and select them using cut-off
- Plot Gain curves and Life curves and interpret results

Business Use Case

As a Data Scientist, you are requested by the hospital to build a predictive model for them. The hospital will apply your predictive model to perform prediction on new patients whether they will become diabetic or not. Further, based on the prediction, the hospital will recommend action plan such as developing a diet plan, physical activities and others.

Step1. [Import dataset]

- Using Pandas, import **"diabetes.csv"** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Identify relationships between feature]

- Create a Heatmap for the dataset and understand the data

Step3. [Prediction using one feature]

Will older people become diabetic?

- Create LogisticRegression model, train with "Age" as **X** and "Outcome" feature as **y**.
- Print model parameter values: coef_ and intercept_
- **Query:** A person is 60 years old. Will he be diabetic?
- Use model parameters and find function value. Your code will be as below.

```
lrf = logreg.coef_ * 60 + logreg.intercept_
from scipy.special import expit
expit(lrf)
```

If your output > 0.5, YES he will become diabetic. Otherwise, NO, he will not be diabetic.

Step4. [Prediction using many features]

Will Glucose, BMI and Age values make someone diabetic?

- Select the three features 'Glucose', 'BMI' and 'Age' from your dataset, call it as **X**
- Create a new LogisticRegression model, train with X and 'Outcome' as **y**.
- Query: For a person, glucose=150, bmi=30, age=40. Will he be diabetic?
- Find the value of expit() as before. Output will be: 0.5208271643241003

Note: You can also verify expit() output value as below

```
logreg.predict_proba([[150, 30, 40]])
```

Step5. [Build LoR model with all features]

- Create LoR model, train it with X_train and y_train values
- Now, compute and print its AUC value
- Can we get this AUC value with limited set of good features?. Yes, we are going to find with 'Forward Selection Procedure'.

Step6. [Forward Selection Procedure]

Now, you have to find and select a good set of features with the best AUC score. The algorithm is

Forward stepwise variable selection procedure

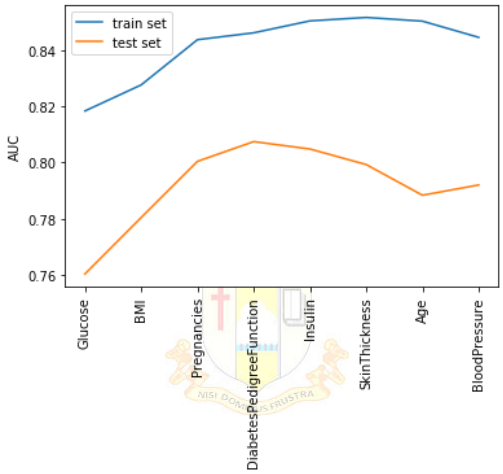
1. Empty set
2. Find best variable v
3. Find best variable v in combination with v
4. Find best variable v in combination with v , v
5. ...
6. (Until all variables are added or until predefined number of variables is added)

Implementation Steps of the forward stepwise procedure

1. Define a function get_auc() that calculates AUC given a certain set of variables
2. Define a function best_next() that returns next best variable in combination with current variables
3. Loop until desired number of variables

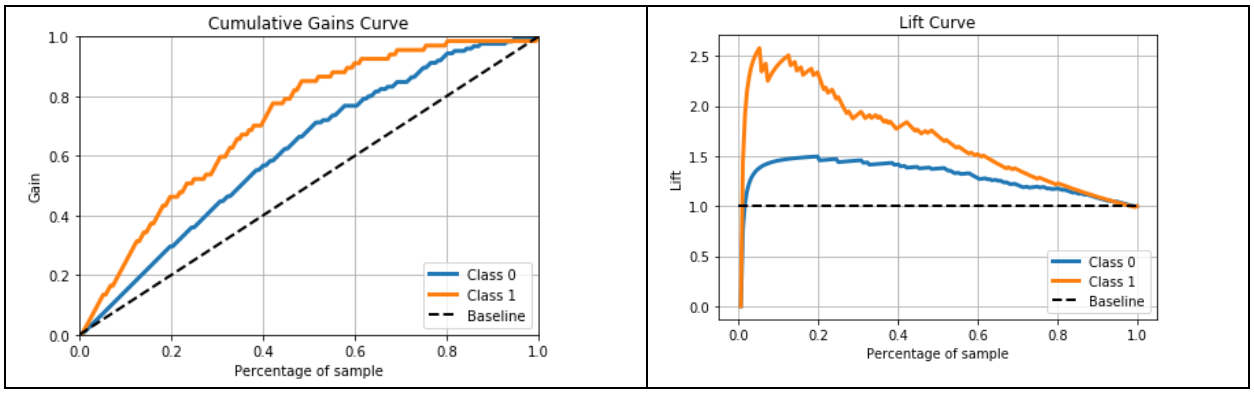
Step7. [Plot Line graph of AUC values and select cut-off]

- Split your dataset equally for training and testing
- Plot AUC values for each variable. The graph will appear as below.



The cut-off line can be drawn at the 4th variable. It gives the best AUC score around 0.81. Therefore, it will be sufficient to use only the first 4 features for training and testing. Otherwise, AUC is going down gradually.

Step8. [Draw Cumulative Gain Chart and Lift Chart]



Interpret these curves.

NOTES

NOTES

NOTES

Lab7. Loan Approval Classification using SVM

Objectives
In this lab, you will build a classification model to classify the loan applicants into eligible applicants or not eligible applicants using Support Vector Machine.

Learning Outcomes
After completing this lab, you will be able to

- Apply data cleaning methods
- Perform EDA and understand who got their loans approved
- Do feature engineering with One Hot Encoding
- Create LinearSVC model, train and predict on the data
- Print accuracy, confusion matrix and classification report
- Compare LinearSVC model with SVC and SGDClassifier models

Business Use Case
Heber Housing Finance deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. However doing this manually takes a lot of time. Hence, it wants you to automate the loan approval process (real time) based on customer information. So you should identify all features and build a model that enable the company to approve the load application or not.

The dataset contains the details of 614 loan applicants, where each applicant is described with 12 features. Loan_Status is the target feature (ie., dependent variable) and all others are independent variables.

Step1. [Understand Data]. Using Pandas, import **"train_loan.csv"** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Data Cleaning]

- Replace numbers as string by integer in "Dependents" column
- Fill missing data in categorical columns (Gender, Married, Dependents, Education, Self_Employed, Credit_History) by its mode value
- Handle missing values in numberical columns
- Drop Loan_ID column

Step3. [OPTIONAL: Exploratory Data Analysis - Who got their loan approved]
Draw count plot for

- Married?
- Dependants?
- Graduates?
- Self-employed?

Step4. [Extract X and y] from the dataframe

Step5. [One Hot Encoding]
Perform OHE on categorical columns, use this method: X = pd.get_dummies(X)

Step6. [Model Building]

- Split X and y for training and testing
- Using StandardScaler, fit_transform on X_train and transform on X_test values
- create LinearSVC model, train and test
- print accuracy value
- Print confusion matrix between y_test and y_pred
- Print classification_report

Step7. [Performance Comparisons]

- Compare the performance of LinearSVC against LogisticRegression
- Compare the performance of LinearSVC against SGDClassifier
- Compare LinearSVC against SVC with various kernels such as ‘linear’, ‘poly’, ‘rbf’ and ‘sigmoid’
- Interpret the results.

NOTES

NOTES

Lab8. Animal Classification using Decision Trees

Objectives
In this lab, you will build ID3 and CART Decision Tree to classify whether an animal is a Mammal or Reptile.

Learning Outcomes
After completing this lab, you will be able to

- Create and import training dataset and test dataset
- Create ID3 Decision Tree using Entropy metric
- Create CART Decision Tree using Gini metric
- Visualize graph using graphviz

Step1. [Create Dataset]
Create the following dataset using Excel and save it as CSV file.

Toothed	hair	breathes	legs	species
TRUE	TRUE	TRUE	TRUE	Mammal
TRUE	TRUE	TRUE	TRUE	Mammal
TRUE	FALSE	TRUE	FALSE	Reptile
FALSE	TRUE	TRUE	TRUE	Mammal
TRUE	TRUE	TRUE	TRUE	Mammal
TRUE	TRUE	TRUE	TRUE	Mammal
TRUE	FALSE	FALSE	FALSE	Reptile
TRUE	FALSE	TRUE	FALSE	Reptile
TRUE	TRUE	TRUE	TRUE	Mammal
FALSE	FALSE	TRUE	TRUE	Reptile

Step2. [Model building using ID3]

- Import your data set
- Create DT model using 'entropy' criterion
- Perform training and testing
- Print accuracy and classification report.
- Interpret your results
- Visualize your DT model using graphviz

Install necessary package for visualization. The following sample code will help you to understand visualization.

```
with open("tree1.dot", 'w') as f:
     f = tree.export_graphviz(dtc_entropy1,
                              out_file=f,
                              max_depth = 4,
                              impurity = False,
                              feature_names = X.columns.values,
                              class_names = ['Reptile', 'Mammal'],
                              filled= True )
```

Note: Once dot file is created, then you can visualize online at http://www.webgraphviz.com/. Open and copy dot file contents and paste it here. You can view your graph.

Step3. [Create a Test Set]
Create a testing csv file with 3 samples as below. (columns 2, 3, 4, 5 only)

Turtile	FALSE	FALSE	TRUE	FALSE	Reptile
Blue whales	FALSE	TRUE	TRUE	TRUE	Mammal
Crocodile	TRUE	FALSE	TRUE	TRUE	Reptile

Step4. [Perform prediction]
Use your ID3 DT model that you created before and predict labels for this test set.
Check your predictions. Correct?

Step5. [Build CART Decision Tree Model]

- Now, you are going to build a new CART decision tree using criterion='gini'.
- Train you model with full training data (No, train test split, this time)
- Predict samples for the test file
- Visualize your CART DT using graphviz

Step6. [Buid DT with Zoo dataset]

- Download full animal dataset at https://archive.ics.uci.edu/ml/datasets/Zoo
- Import, build model using ID3 and CART, train and test accuracy. Print classification report. Visualize your trees.

NOTES

NOTES

NOTES

Lab9. Employee Hopping Prediction using Random Forests

Objectives

In this lab, you will build a classification model to predict whether employee will continue to work or quit his job in the company using Random Forest classifier.

Learning Outcomes

After completing this lab, you will be able to

- Perform One Hot Encoding on categorical columns
- Create RandomForestClassifier, perform training and testing
- Print accuracy score and classification report
- Print feature importance values
- Select the best number of trees based on out-of-bag error values
- Compare against Decision Tree model
- Visualize trees using graphviz

Business Use Case

You are a data scientist. Heber Software Solutions is a leading IT industry in your city. They have collected the details of employees and if they work or left the company. They ask you to build a prediction model so that they can use your model to check if employees will continue to work or leave. Based on this analysis, they will understand what make them to quit and accordingly they will design employee welfare management schemes.

Step1. [Understand Data].

- Using Pandas, import **"Employee_Hopping.csv "** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Extract X and y]

- Create X and y columns from the dataframe

Step3. [Feature Engineering]

- There are 8 categorical columns (where dtype="object"). Perform one hot encoding and create new columns

Step4. Now, check shape of X and y.

Step5. [Model Development]

- Split X and y for training and testing
- Create RandomForestClassifier model, fit (no need to scale) and predict

Step6. [Testing]

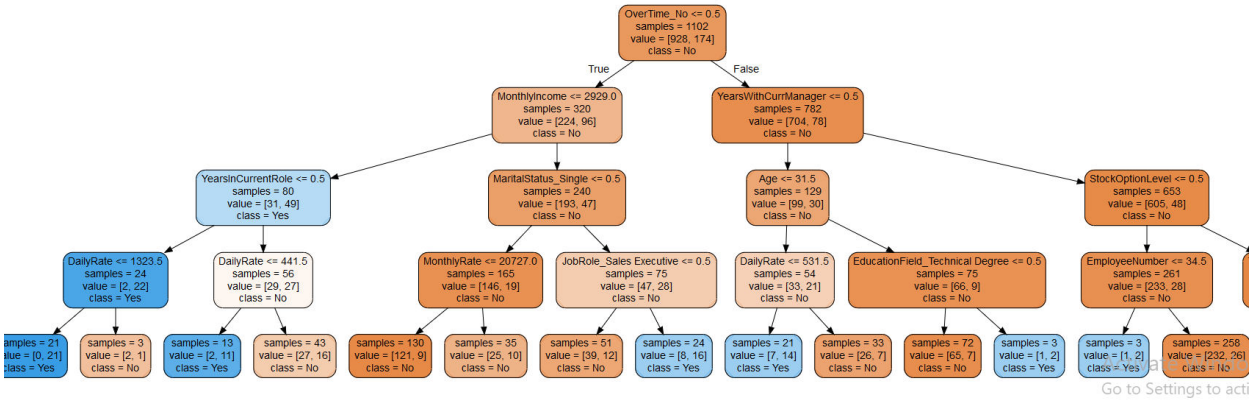
- Print accuracy score between y_test and y_pred
- Print classification report between y_test and y_pred and observe the results

Step7. [Feature importance value]

- You can look at feature importance values using the property, rf.feature_importances_
- Print feature name and its rf.feature_importances_ values and understand important features
- Show a Bar plot between feature column names and feature_importances_ score.

Step8. [Visualize your RF Decision Tree using graphviz]

- Figure below show a segment of the RF tree which is visualized at http://www.webgraphviz.com/. You should copy and paste .dot file in this page.



Step9. [RF with a range of trees]

Fit random forest models with a range of tree numbers [15, 20, 30, 40, 50, 100, 150, 200, 300, 400] and print Out-Of-Bag error for each of these model. Use model.oob_score_ to get score and subtract this score from 1 to get the oob-error. That is, oob-error = 1 - model.oob_score_.

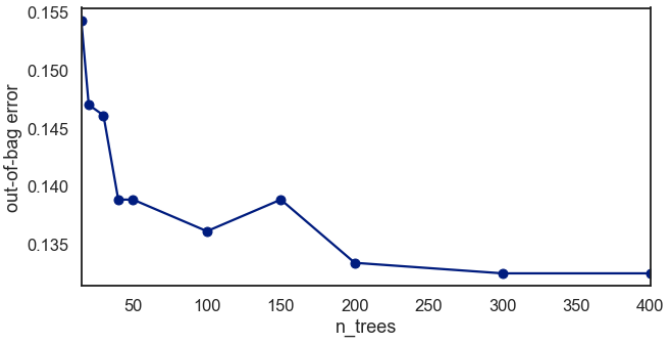
Hint: since the only thing changing is the number of trees, the `warm_start` flag can be used so that the model just adds more trees to the existing model each time. Use the `set_params` method to update the number of trees. The following code many help to understand this part.

```
rf2 = RandomForestClassifier(oob_score=True,
                             random_state=42,
                             warm_start=True,
                             n_jobs=-1)

oob_list = list()

# Iterate through all of the possibilities for number of trees
for n_trees in [15, 20, 30, 40, 50, 100, 150, 200, 300, 400]:

    # Use this to set the number of trees
    rf2.set_params(n_estimators=n_trees)

    # Fit the model
    rf2.fit(X_train, y_train)

    # Get the oob error
    oob_error = 1 - rf2.oob_score_

    # Store it
    oob_list.append(pd.Series({'n_trees': n_trees, 'oob': oob_error}))

rf_oob_df = pd.concat(oob_list, axis=1).T.set_index('n_trees')

rf_oob_df
```

Step10. [Plot oob-error for each tree]

The following lines will help you

```
ax = rf_oob_df.plot(legend=False, marker='o', figsize=(10,5))
ax.set(ylabel='out-of-bag error')
```



Step11. [Compare with DecisionTreeClassifier]

- Create DecisionTreeClassifier, fit and predict on test set
- Visualize the tree using graphviz
- Print accuracy score
- Print classification report
- What is the result of the comparision between RF and DT models? Which gives best accuracy?.
- What is your comment on precision, recall, f1 score values?

NOTES

NOTES

NOTES

Lab10. Patients Physical Activities Prediction using Boosting

Objectives

In this lab, you will recognize physical activities such as 'laying', 'sitting' or 'walking' using Gradient Boosting, AdaBoost and VotingClassifiers.

Learning Outcomes

After completing this lab, you will be able to

- Create a small dataset with selected rows based on fewer target labels
- Build GradientBoostingClassifier, fit and predict on test data
- Print accuracy and classification report
- Find the best no. of decision trees and learning rate using GridSearch and Cross Validation
- Build AdaBoost classifier model with GridSearchCV, fit and predict
- Select best parameter values for n_estimators and learning_rate
- Build LogisticRegressionCV model, fit, predict and print scores
- Build VotingClassifier using other models, fit, predict and print scores
- Interpret results and parameter values
- Change parameter values and play around with models

Business Use Case

As you a data scientist, one popular hospital in your city has asked you to design a model based on the mobile phone data of their patients. Your model will help the hospital to understand the physical activity level of their patients. The physical activities are classified into 3 levels – Laying, Sitting and Walking. Based on the prediction, Doctors will recommend health diets, exercise and physical activities to the new patients.

Dataset

You will use a dataset named, ***Human_Activity_Data.csv***.

Step1. [Understand Data]

- Using Pandas, import ***"Human_Activity_Data.csv"*** file and print properties such as head, shape, columns, dtype, info and value_counts.

Step2. [Build a small dataset]

- As it is a big dataset, execution time will be long for training and testing. So build a small dataset with only 3 classes, laying, sitting and walking, where each class with 500 samples. So, shape of this new dataframe will be (1500, 563). That is 1500 rows and 563 features.
- Store this new dataframe as a CSV file.

Step3. [Build GradientBoostingClassifier]

- Import your new reduced CSV file
- Print basic properties of the new CSV file
- Split it into training set and test set (30% samples for testing)
- Create GradientBoostingClassifier, fit and predict
- Print accuracy and classification report

Step4. [Find Best no. of trees and Best Learning Rate using Grid Search and Cross Validation]

- Create GridSearchCV model with GradientBoostingClassifier
- Parameters: *param_grid = {'n_estimators': [50, 100, 200, 400], 'learning_rate': [0.1, 0.01]}*
- Perform fit and predict
- Print accuracy, classification report
- Print best parameters such as best no. of trees and learning rate. Use the attribute best_estimator_

Step5. [Build AdaBoostClassifier]

- Create AdaBoostClassifier with DecisionTreeClassifier
- Create GridSearchCV with AdaBoostClassifier model that you created as before
- Parameters: *param_grid = {'n_estimators': [100, 150, 200], 'learning_rate': [0.01, 0.001]}*
- Perform fit, predict
- Print accuracy, classification report
- Print best parameters such as best no. of trees and learning rate. Use the attribute best_estimator_

Step6. [Build LogisticRegressionCV classifier]

- Create a LogisticRegressionCV model with the parameters Cs=5, cv=4, penalty='l2'.
- Perform fit and predict
- Print classification report

Step7. [Build VotingClassifier]

- Build VotingClassifier model with GradientBoostingClassifier and LogisticRegressionCV that you created in the previous steps
- Perform fit and predict operations
- Print classification report

Step8. [Interpret your results]

- Analyze your results
- Change parameters and play with your code

NOTE

NOTE

NOTES

Lab11. Shopping Mall Customer Segmentation using Clustering

Objectives

In this lab, you will detect clusters from customer data and perform analytics to understand customers who visit malls, using KMeans and Agglomerative Clustering methods.

Learning Objectives

After completing this lab, you will be able to

- Perform Skew analysis and interpretation
- Check if data normalization is required
- Build KMeans model with the required no. of clusters
- Visualize clusters based on selected features
- Select the best value for K using inertia error values and Elbow method
- Perform Cluster Analysis to understand cluster statistics
- Reduce dimensions of data using PCA and build KMeans model
- Build MeanShift clustering model on dimensions reduced data
- Create hierarchical clusters using Agglomerative Clustering
- Visualize hierarchical clusters using Dendrogram

Business Use Case

You are given the data of customers who have visited your mall. The details of customers such as age, annual income, spending score and gender are collected for each customer. They have asked you to analyse this data and give insights. This will help them to design promotion strategies, marketing models and others so as to reach out specific group of customers. Also, those group of customers will be targeted via surveys to collect further details. Based on that feedback we can decide whether the new strategy is good for that customer segment or not, even before the strategy is released.

Step1. [Understand Data]

- Using Pandas, import ***"Mall_Customers.csv"*** file and print properties such as head, shape, columns, dtype, info and value_counts.
- For example: **customers_data = pd.read_csv("Mall_Customers.csv")**

Step2. [Label encode gender]

- Genre (ie., gender) is a string, so label encode into binary

Step3. [Check for variance]

- Use describe() on your data frame and check for variance. If variance is high for float columns, you need to normalize. Otherwise, ignore

Step4. [Check skewness]

- Check if float columns are skewed. Use skew() on your data frame. If skew value is greater than 0.75, then you can perform log transformation on those skew columns.

Step5. [Pair plot]

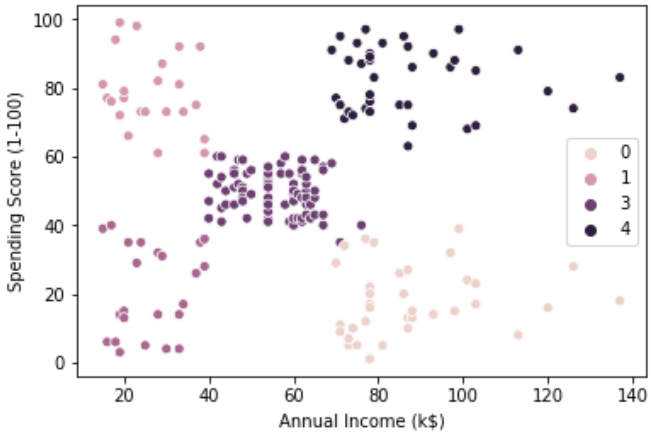
- Draw pair plot and observe correlations.

Step6. [Build KMeans]

- Create and fit KMeans (n_clusters variable can be set with any value)
- Print label_ and cluster_centers_ values

Step7. [Scatter plot]

- Draw scatter plot between any two features with hue as "labels_". This figure shows 5 clusters.



Step8. [Cluster Analysis].
Now, predict cluster labels for the same data. For example,

```
kmeans2 = KMeans(n_clusters = 5, init='k-means++')
kmeans2.fit(customers_data)
pred = kmeans2.predict(customers_data)
```

Now, add a new column for pred in a new dataframe, such as

```
frame = pd.DataFrame(customers_data)
frame['cluster'] = pred
```

This will create a new column to frame. That means, you have added a cluster prediction column, whose values say the cluster number to which the row belongs to. That is, that customer belongs to that cluster number.

Now, group customers based on cluster number. Remember, here we have 5 clusters from 0 to 4.

For each cluster group, print the following details.

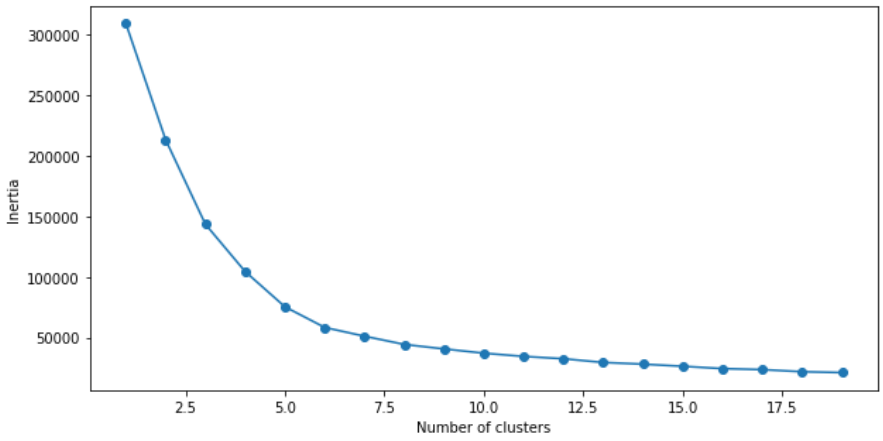
```
Average age:  45.21739130434783
Average annual income:  26.304347826086957
Deviation of the mean for annual income:  7.893811054517766
No of customers ie shape:  (23, 5)
From those customers we have 9 male and 14 female
```

Step9. [Find the best number of clusters]

Compute inertia value as shown below

```
SSE = []
for clust in range(1,20):
    km = KMeans(n_clusters=clust, init="k-means++")
    km = km.fit(customers_data)
    SSE.append(km.inertia_)
```

Plot a line chart between cluster number and its inertia value. You will get graph as below. Can you identify the best number of clusters from this graph?

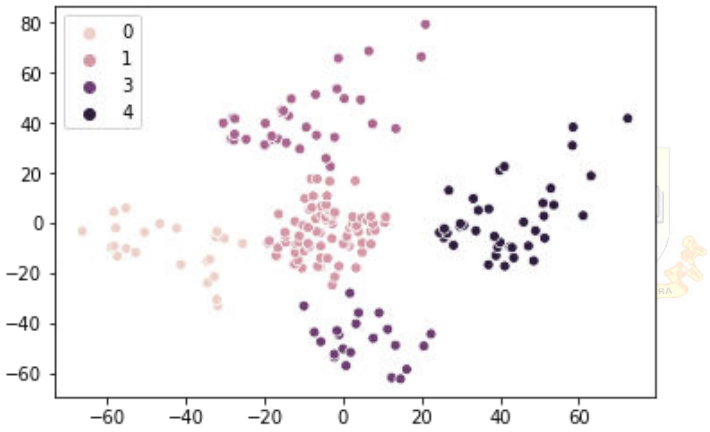


Step10. [Reduce Dimensions using PCA]

- Reduce 4 dimensions into 2 dimensions using PCA
- Create KMeans model, fit on the reduced dataset
- Print cluster_centers_ and labels_

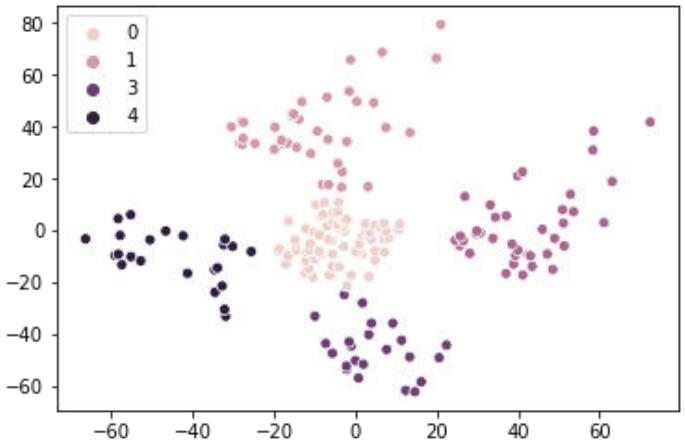
Step11. [Scatter plot]

- Draw a scatter plot between the 2 reduced dimensions, with hue as label_
- Your scatter plot may look like below



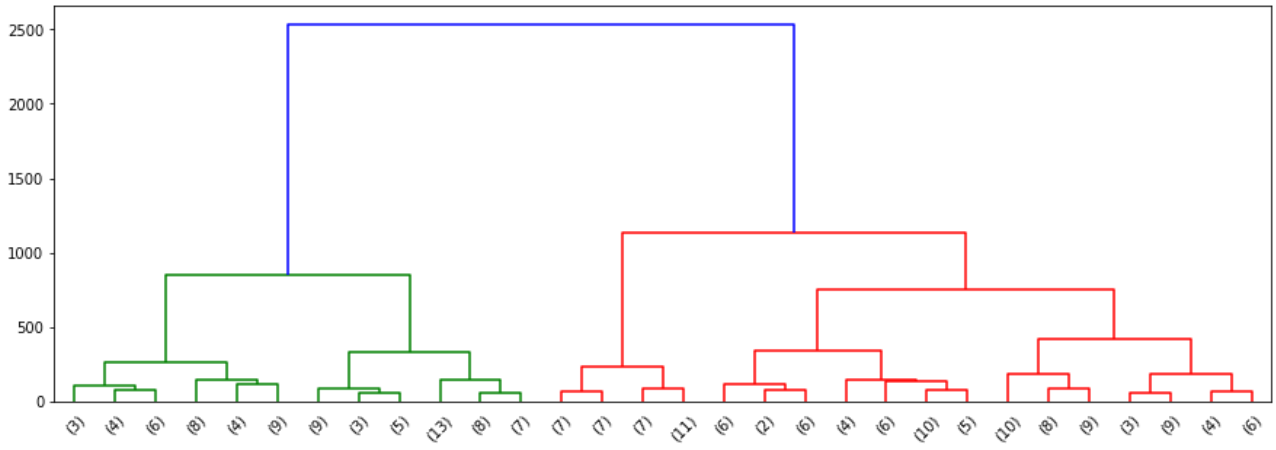
Step12. [MeanShift clustering]

- Create MeanShift clustering model and fit on the reduced data of PCA and visualize clusters on the reduced data, as shown below.



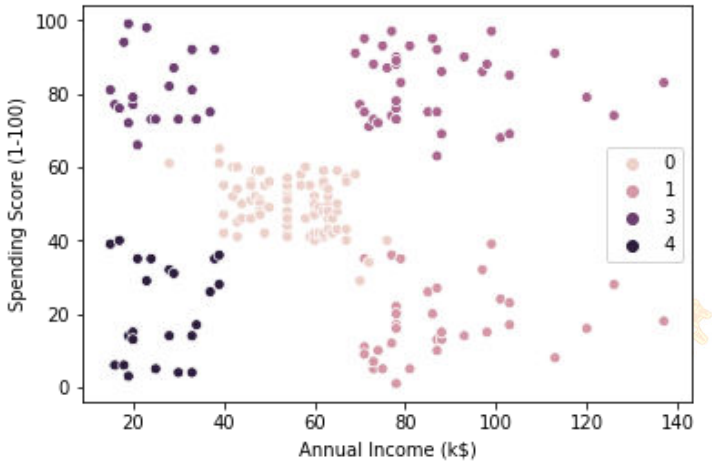
Step13. [Predict hierarchical clusters using AgglomerativeClustering]

- Create 5 clusters, using AgglomerativeClustering class. Your dendrogram will look like as below.



Step14. [Visualize scatter plot with hue as agglomerativeclustering labels_]

- Visualize agglomerative clusters using the predicted label. Select any two features for X and Y with hue as labels_. Your scatter plot will look like below



NOTE

NOTES

NOTES