**Python basics**

**Strings**

How to append a str into a column of pandas dataframe.

df['col'] = 'str' + df['col'].astype(str)

uses: apply the ‘str’ to all rows in the column specified

-------------------------------------------------------------------------

**Tuples**

Tuple is just like list but different in the way they are written such as using ( ) instead of [ ].

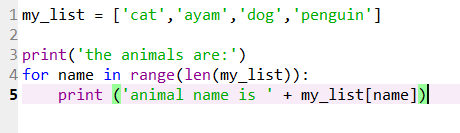
Tuple is also immutable like string meaning cannot make changes in the value.

**Lists**

List = [‘a’,’b’,’c’]

List is mutable meaning can change the value while string is immutable (\*unless use concatenation)

How to print out the value of each data in a list:-



**Machine Learning**

**Supervised Learning**

**1. Basic imports:**

Pandas as pd

Numpy as np

Matplotlib.pyplot as plt

Seaborn as sns

**2. Load\_data:**

df = pd.read\_csv(‘filename.csv’)

METHOD 1

\*\*\*assign dependent and independent variables\*\*\*

X = df[‘columnname1’ , ‘columnname2’ , … ]

Y = df[‘labelledcolumn’]

METHOD 2

X = data.iloc[ row , column ]

-----------------------------------------------------------------

**3. Explore data analysis [do visualization]:**

Plot a graph (scatterplot)

Plt.scatter(X, y, color=’’)

Plt.xlabel(‘Xlabelname’, fontsize=??)

Plt.ylabel(‘ylabelname’, fontsize=??)

Plt.show()

**Cheat sheet for Machine learning:**

print('Misclassified samples: %d' % (y\_test != y\_pred).sum()) #to check total of wrong prediction

**DUMMY VARIABLE TECHNIQUES:**

To change categorical value in data to numerical value

Example:

Dataframe[‘columnname’] = dataframe[‘columnname’].map({‘Attr1’ : val, ‘Attr2’ : vagl})

**Exploratory Data Analysis (EDA)**

Involve in doing visualization to your data. Some of the libraries use are pandas and seaborn as they have built-in plotting capabilities.

**SIMPLE LINEAR REGRESSION**

Equation of simple linear regression(SLR) :

*Y = mX + c*

Y : dependent variable

X : independent variable

m : Coefficient ( multiplier that relate X to Y )

c : constant ( y- intercept )

**How to find the best fit line using SLR?**

**MULTIPLE LINEAR REGRESSION**

Equation of simple linear regression(SLR) :

*Y = C + mX + m2X2 + m3X3 + m4X4 …. mnXn*

Y : dependent variable (something we try to explain)

X : independent variable

m : Coefficient ( multiplier that relate X to Y )

c : constant ( y- intercept )

NOTE : There’s a CATEGORICAL variable and NUMERICAL variable.

Approach for categorical variable is to create a dummy variable.

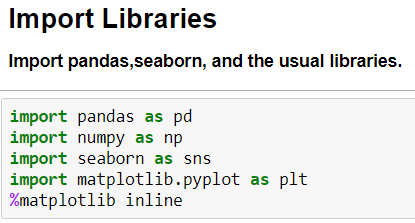
**LOGISTIC REGRESSION**

**K Nearest Neighbors**

K Nearest Neighbors or KNN is a model that is used to solve classifying problem with unknown features, target.

**Steps for KNN**

1. Importing libraries



1. Read your data file(csv,excel,db) to DataFrame.

* (\*anyname) = pd.read\_csv(‘\*datafilename’)

1. Explore data analysis (EDA)

* You can use seaborn to explore and visualize data to see any relationships between the features. For more info on how to do **EDA**, refer to **EDA** section.

1. Standardize the variables/features

If the value between features too far apart.

Import from scikit learn and create object for standardscaler()

* from sklearn.preprocessing import StandardScaler
* scaler = StandardScaler()
* scaler.fit(df.drop(‘TARGET CLASS’,axis = 1) \*drop target column, just pass the features
* scaled\_feature = scaler.transform(df.drop(‘TARGET CLASS’),axis=1))
* df\_feat = pd.DataFrame(scaled\_features,columns=df.column[:-1]) \*put scaled feature to dataframe

1. Split your data into training and testing/validation set

* From sklearn.cross\_validation import train\_test\_split
* X = df\_feat \*X is your actual features list
* y= df[‘TARGET CLASS’]
* X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=101)

1. Import KNN from scikit learn

* From sklearn.neighbors import KNeighborsClassifier
* knn = KNeighborsClassifier(n\_neighbors=1)
* knn.fit(X\_train, y\_train)

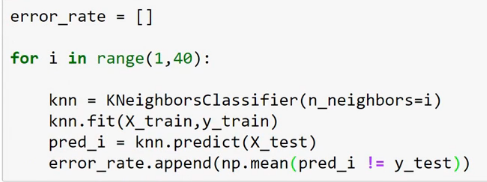
1. Predictions and Evaluations

* pred = knn.predict(X\_test) \*the predict method to predict values using KNN and X\_test

1. Create confusion matrix and classification report

* from sklearn.metrics import classification\_report,confusion\_matrix
* print(confusion\_matrix(y\_test,pred))
* print(classification\_report(y\_test,pred))

1. Use elbow method to pick a good K Value



* plt.figure(figsize = (10,6))
* plt.plot(range(1,40), error\_rate, colors=’blue’, linestyle=’—‘, marker = ‘o’, markerfacecolor = ‘red, markersize= 10)
* plt.title(‘Error rate vs K’)
* plt.xlabel(‘K’)
* plt.ylabel(‘Error Rate’)

1. Retrain with new K Value

* knn = KNeighborsClassifier(n\_neighbors=30)
* knn.fit(X\_train, y\_train)
* pred = knn.predict(X\_test)
* Do step 8

**K-means Clustering**

Identifying certain groups (label) in a dataset.

Steps in K-Means

1. Choose the no. K of clusters.
2. Select random K points, the centroids on the scatter plot.
3. Assign each data point to the closest centroid
4. Compute and place the new centroid
5. Reassign each data to new centroid.

How to know the optimal number of K cluster

-Use WCSS vs K numbers (The Elbow Method)

Then after get the Optimal K value.. use it in KMeans(n\_clusters = X)

**Support Vector Machine (SVM)**

**Tree Methods and Random Forest**

**NATURAL LANGUAGE PROCESSING (NLP)**

Main library of NLP:

Natural language Toolkit – NLTK

Spacy

Stanford NLP

OpenNLP

Bag of words MODEL

-vocab of known words

-a measure presence of known words

Steps.

1. Import usual library – numpy,pandas,matplotlib
2. Import dataset

dataset = pd.read\_csv('Restaurant\_Reviews.tsv', delimiter= '\t', quoting = 3)

\*to check for duplicates in text

\*preprocess the texts as in no punctuation, to lowercase and other feature that u needed

1. checkUnique = dataset.groupby(‘LABEL’).describe() ….

import string

mention = 'Toyota car is impressive! and also, it is cheap than other brands: Honda, Nissan'

nopunc = [p for p in mention if p not in string.punctuation]

nopunc = "".join(nopunc)