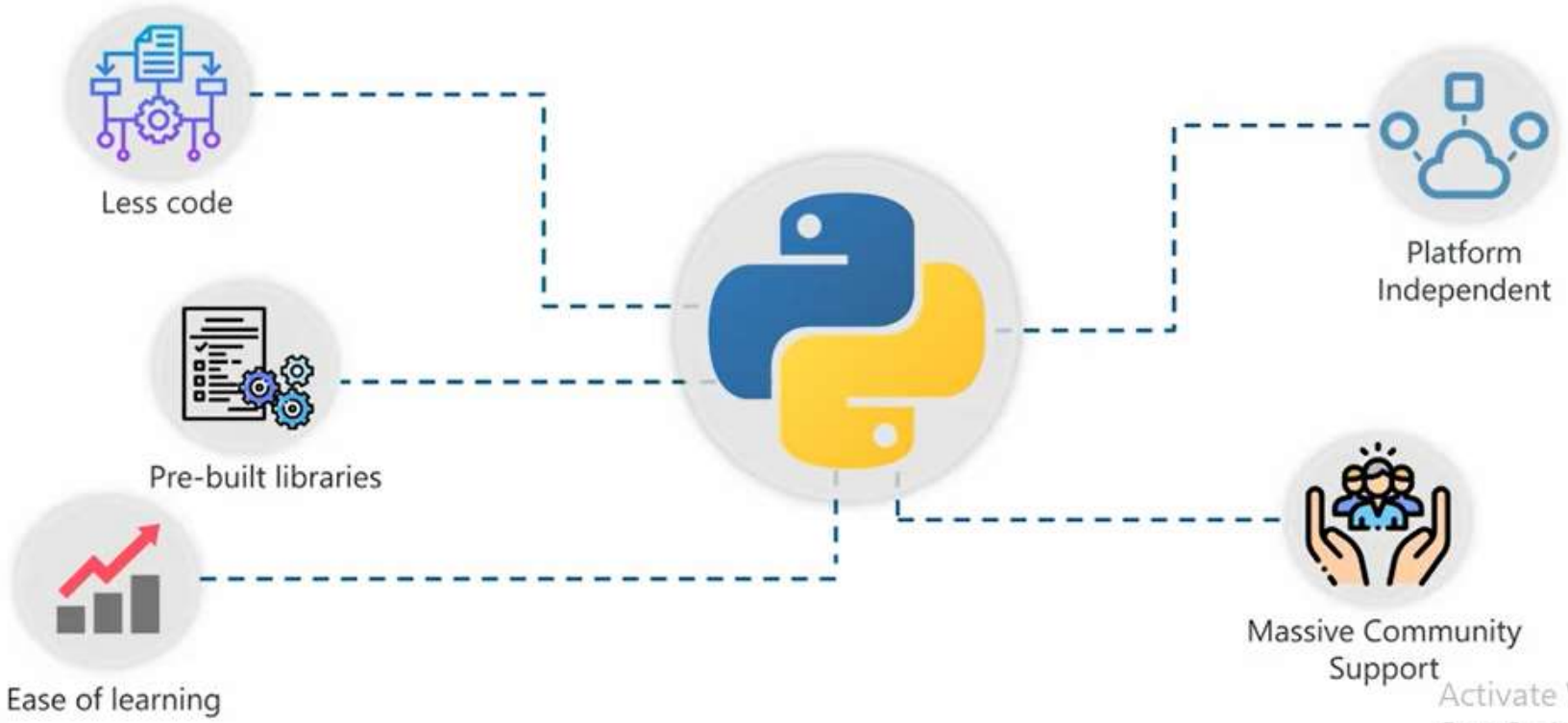
AI – Machine Learning Developer course

Ashok Kumar S
Assistant Professor/CSE,
MSAJCE

Why to use Python in ML?



Python libraries for statistical analysis

- Numpy
- Pandas
- Scipy
- Statsmodels

Numpy(Numerical python)

- Supports operations on multidimensional array objects
  - Collection of methods to process the array elements
    - Used for indexing, sorting
  - For mathematical/any type of logical operations
  - Linear algebra computations like Linear regression, Logistic regression, Naïve bayes etc.,

Pandas

- Used in statistics, finance, economics, data analysis
- Depends on numpy arrays for processing pandas data objects.
- Numpy, Scipy and Pandas are dependent on each other for performing scientific computation or to perform data manipulation
- Used to process/manipulate large data sets and performing data slicing, subsetting etc
- It can create data frame objects with predefined and customized indexing
- It also helps in performing any kind of complex data analysis(Eg: Descriptive statistical analysis, Data wrangling, transformation)
- It provide support for manipulating of time series data

Scipy

- Scipy is built on top of numpy
- Provides a collection of sub packages which helps in solving the basic problems related to statistical analysis
- Used to process array of elements defined using numpy library. It is often used to compute mathematical equations that cannot be done by numpy.
- It is used in solving numerical integration, fourier transform
- Scipy is used in clustering(Kmeans algorithm), signal processing, creating sparse matrices

Statsmodels

- It is built on top of numpy and scipy
- It is a best library to perform statistical testing and hypothesis testing which are not found in numpy/scipy
- Used to implement generalised linear models(Linear Regression, Logistic Regression)

Python libraries for data visualization

- Data visualization includes the use of charts, graph, mind map, heat map, histogram, density plot to study the correlation between the various data variables

- Matplotlib
- Seaborn
- Plotly

Matplotlib

- Support for histogram, barchart, scatterplot
- 2-D graphical library which produce clear graphs important for EDA
- It contains pyplot module that provides basic interface which is similar to matlab UI
- Most used library

Seaborn

- Matplotlib forms the base for seaborn library
- Seaborn is used to create more descriptive statistical graphs
- It comes with inbuilt API for studying relationship b/w multiple vars
- It helps in analyzing and visualizing univariate & bivariate data points.
- It supports automated statistical estimation and graphical representation of Linear Regression models for different types of target vars(Continous/Categorical)
- Supports built-in themes for styling & for creating matplotlib graphs

Plotly

- It supports collection of chart types which includes all 3 D charts.
- It can be used to build public/private dashboard using API
- The visualizations created using plotly is stored in json format, so these visualizations can be accessed in different platforms R, Matlab, Julia etc
- It also comes with inbuilt API called plotly grid which allows us to import data directly into the plotly environment(i.e) we need to import & visualize the var to understand which var is important

Top python libraries for ML

- Scikit learn
- XGBoost

Scikit learn

- Best library for data modeling and model evaluation
- It contains all supervised & unsupervised learning algorithms & also comes with well defined functions for ensemble functions and boosting ML
- It provides set of standard datasets to start with ML(Iris, boston house prices dataset etc)
- It has inbuilt functions to carry out both Supervised & Unsupervised learning
- It has inbuilt functions to perform feature extraction & feature selection which helps in identifying significant attribute/features/variables in data
- It also comes with function for cross validation so that we can estimate the performance of model.
- It also comes with parameter tuning to improve the model performance

XGBoost(Extreme Gradient Boosting)

- One of the best library for boosting ML
- It is written in C++ & is considered as fastest & effective library to improve the performance of the ML model
- The core XGBoost algorithm is parallelizable & it can use the power of multicore computers. This makes this library to process huge datasets & to work across the network of datasets
- It also helps in performing cross validation, parameter tuning, regularization, handling missing values

Working with pandas

- Display the string He is a good person

string
He
is
a
good
person

Implementation

```
import pandas as pd
string=[“He",“is",“a“,”good”,”person”]
sentence=pd.DataFrame(string)
print(sentence)
```

Working with pandas

- Display the list of cars and their corresponding price

Cars	Price
BMW	30000000
Ford	2000000
Indica	500000

Implementation

```
import pandas as pd
car_data={'cars':["BMW","Ford","Indica"],'price':[30000000,2000000,500000]}
list_cars=pd.DataFrame(car_data)
print(list_cars)
```

- Display the name of the cars alone
- print(list_cars[‘cars’])
- Display the details of 2nd car
- print(list_cars.loc[1])

Adding a new data to the existing data frame

```
    cars      price
0   BMW     30000000
1   Ford     2000000
2   Indica    500000
```

```
newdata={'cars':"Inova",'price':1000000}
list_cars=list_cars.append(newdata,ignore_index=True)
print(list_cars)
```

```
    cars      price
0   BMW     30000000
1   Ford     2000000
2   Indica    500000
3   Inova    1000000
```

Updation of data in the data frame

```
    cars     price
0   BMW    30000000
1   Ford     2000000
2   Indica   500000
3   Inova    1000000
```

```
list_cars.at[2,'price']=450000
print(list_cars)
```

```
    cars     price
0   BMW    30000000
1   Ford     2000000
2   Indica    450000
3    Inova    1000000
```

```
list_cars.iloc[[1,3],[1]]=5000000   =>  Change the index 1 and 3 and column 1 to 50L
    cars     price
0   BMW    30000000
1   Ford     5000000
2   Indica    450000
3    Inova   5000000
```

Changing of index for the dataframe

```
list_cars.index=[1,2,3,4]
print(list_cars)
```

```
cars          price
1   BMW      30000000
2   Ford      5000000
3   Indica     450000
4  Inova     5000000
```

Storing and retrieving the dataframes

storing of data frame to csv file

```
list_cars.to_csv('carpriceinfo.csv')
```

Reading a csv file and storing in a dataframe

```
carinformation=pd.read_csv('carpriceinfo.csv')
print(carinformation)
```

Other operations

```
car_price={'cars':["BMW","Ford","Indica"],'price':[30000000,2000000,500000]}
df1=pd.DataFrame(car_price)
newdata={'cars':["Inova","Audi","Honda"],'price':[1000000,500000000,10000000]}
df2=pd.DataFrame(newdata)
df1=df1.append(df2,ignore_index=True)

print(df1)
print(df1.head())
print(df1.head(3))
print(df1.tail(2))
print(df1.shape)
print(df1.describe())
print(df1.info())
print(pd.__version__)
```

Deletion of data in dataframe

```
list_cars=list_cars.drop(list_cars.index[1])
print(list_cars)
```

```
list_cars=list_cars.drop(list_cars.index[1:3],inplace=True)
print(list_cars)
```

Deletion of columns in dataframe

```
list_cars=list_cars.drop(['cars'],axis=1)
print(list_cars)
```

Working with pandas

Day	Calories consumed	Calories burnt
Mon	2500	300
Tues	2340	450
Wed	1750	400
Thurs	2600	600
Fri	3300	350
Sat	2250	380
Sun	2800	500

- Display the list calories consumed and burnt on each day
- Add an additional column calories remining

[Note: calories remining = calories consumed - calories burnt]

Exercise

- Create a same dataframe calories consumed and calories burnt for all the days from Monday to Sunday.
- The days should not be a separate column. Days should act as a index to the dataframe.

	Calories consumed	**Calories burnt**
Mon	2500	300
Tue	2340	450
Wed	1750	400
Thu	2600	600
Fri	3300	350
Sat	2250	380
Sun	2800	500