2024

Project Report

SHRIYA SINGH

NIELIT CHANDIGARH

CONTENTS

WHAT IS PYTHON?

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

- web development (server-side),
- software development,
- mathematics,
- system scripting.

BUILT-IN DATA TYPES

IN PROGRAMMING, DATA TYPE IS AN IMPORTANT CONCEPT.

VARIABLES CAN STORE DATA OF DIFFERENT TYPES, AND DIFFERENT TYPES CAN DO DIFFERENT THINGS.

Python has the following data types built-in by default, in these categories:

Text Type:	`str`
Numeric Types:	`int`, `float`, `complex`
Sequence Types:	`list`, `tuple`, `range`
Mapping Type:	`dict`
Set Types:	`set`, `frozenset`
Boolean Type:	`bool`
Binary Types:	`bytes`, `bytearray`, `memoryview`

None Type: `NoneType`

GETTING THE DATA TYPE

You can get the data type of any object by using the `type()` function.

PYTHON NUMBERS

There are three numeric types in Python:

- `int`
- `float`
- `complex`

INTRODUCTION TO PANDAS

• pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, built on top of the Python programming language.

•It aims to be the fundamental high-level building block for doing practical, real-world data analysis in Python.

•It is the most powerful and flexible open source data analysis/manipulation tool available in any language.

• It contains data structures and data manipulation tools designed to make data cleaning and analysis fast and easy in Python.

•pandas is often used in tandem with numerical computing tools like NumPy and SciPy, analytical libraries like statsmodels and scikit-learn, and data visualization libraries like matplotlib.

•pandas adopts significant parts of NumPy's idiomatic style of arraybased computing, especially array-based functions and a preference for data processing without for loops.

•While pandas adopts many coding idioms from NumPy, the biggest difference is that pandas is designed for working with tabular or heterogeneous data. NumPy, by contrast, is best suited for working with homogeneous numerical array data.

PANDAS DATA STRUCTURES

The two primary data structures of pandas:

•Series (1-dimensional) and

•DataFrame (2-dimensional), handle the vast majority of typical use cases in finance, statistics, and many areas of engineering.

DIFFERENT WAYS OF CREATING DATAFRAME

1.df=pd.read_csv(“weather_data.csv”)

2. df=pd.read_excel(“wether_data.xlsx”, “Sheet1”)

3. Through dictionary

weather_data={‘day’:[‘01-09-2016’, ’01-10-2016’, ‘1/16/2016’ , ’1/27/2016’],‘temperature’ :[32,34,45,23],‘windspeed’ : [6,7,4,3]}

df=pd.DataFrame(weather_data)

READING/WRITING CSV FILES

skiprows=1 header = None # if columns headers are missing in csv

names=[List of column names]

nrows=3 # to fetch few rows

na_values = [‘not available’, ‘n.a.’]

na_values = {‘eps’ : [‘not available’ , ‘n.a.’],‘revenue’: [‘not available’, ‘n.a.’, -1]}

DATAFRAME FUNCTIONS

fillna – to fill missing values

interpolate – to make a guess on missing values using interpolation

dropna – to drop rows with missing values

SELECTING MULTIPLE ROWS AND COLUMNS

from a pandas dataframe

df.loc ⍰ It selects row/columns based on labels

df.loc[0, :] – 0

th row and all columns

df.loc[[0,1,2], :] - 3 rows and all columns

df.loc[0:2, :] - 3 rows and all columns # 0 and 2 are inclusive

df.loc[0:2] - You can skip columns value if need all columns

GROUPING

```
g=df.groupby('city')
for city, city_df in g:

print(city)

print(city_df)

g.get_group('mumbai') # similar to sql grouping

g.max()
```

PYTHON LIBRARIES FOR DATA SCIENCE

Many popular Python toolboxes/libraries:

- NumPy
- SciPy
- Pandas
- SciKit-Learn

Visualization libraries

- matplotlib
- Seaborn

and many more …

SciKit-Learn:

- provides machine learning algorithms: classification, regression, clustering, model validation etc.
- built on NumPy, SciPy and matplotlib

matplotlib:

- python 2D plotting library which produces publication quality figures in a variety of hardcopy formats
- a set of functionalities similar to those of MATLAB
- line plots, scatter plots, barcharts, histograms, pie charts etc.
- relatively low-level; some effort needed to create advanced visualization

INTRODUCTION TO PANDAS DATA STRUCTURES

- import pandas as pd
- from pandas import Series, DataFrame

To get started with pandas, you will need to get comfortable with its two workhorse data structures: Series and DataFrame. While they are not a universal solution for every problem, they provide a solid, easy-to-use basis for most applications.

SERIES

- Often it will be desirable to create a Series with an index identifying each data point with a label:

- In [15]: obj2 = pd.Series([4, 7, -5, 3], index=['d', 'b', 'a', 'c'])
- d 4
- b 7
- a -5
- c 3
- dtype: int64
- In [17]: obj2.index
- Index(['d', 'b', 'a', 'c'], dtype='object')

NAN OBJECT

- When you are only passing a dict, the index in the resulting Series will have the dict's keys in sorted order. You can override this by passing the dict keys in the order you want them to appear in the resulting Series:
- states = ['California', 'Ohio', 'Oregon', 'Texas']
- obj4 = pd.Series(sdata, index=states)
- obj4

```
California        NaN

Ohio                35000.0

Oregon           16000.0

Texas              71000.0 dtype: float64
```

Here, three values found in sdata were placed in the appropriate locations, but since no value for 'California' was found, it appears as NaN (not a number), which is considered in pandas to mark missing or NA values. Since 'Utah' was not included in states, it is excluded from the resulting object.

- The isnull and notnull functions in pandas should be used to detect missing data:
- pd.isnull(obj4)

```
California True

Ohio          False

Oregon     False

Texas         False

dtype: bool
```

- pd.notnull(obj4) Out[33]:
- California   False

```
Ohio            True

Oregon       True

Texas          True

dtype: bool
```

- Series also has these as instance methods:
- obj4.isnull()

SERIES ALIGNMENT

- A useful Series feature for many applications is that it automatically aligns by index label in arithmetic operations:
- obj3

```
Ohio         35000

Oregon    16000

Texas        71000

Utah           5000

dtype: int64
```

- obj4

```
California        NaN

Ohio                  35000.0

Oregon           16000.0

Texas                71000.0
```

- obj3 + obj4

```
California          NaN

Ohio                70000.0

Oregon         32000.0

Texas            142000.0

Utah                NaN
```

DATAFRAME

- A DataFrame represents a rectangular table of data and contains an ordered collection of columns, each of which can be a different value type (numeric, string, boolean, etc.).

- The DataFrame has both a row and column index; it can be thought of as a dict of Series all sharing the same index.
- Under the hood, the data is stored as one or more two-dimensional blocks rather than a list, dict, or some other collection of one-dimensional arrays

RETRIEVING COLUMNS IN A DATAFRAME

- A column in a DataFrame can be retrieved as a Series either by dict-like notation or by attribute:
- frame2['state']
- frame2.year
- frame2.loc['three']
- Columns can be modified by assignment. For example, the empty 'debt' column could be assigned a scalar value or an array of values:
- frame2['debt'] = 16.5
- frame2['debt'] = np.arange(6.)

ASSIGNINNMENT IN A DATAFRAME

- Val=pd.Series([-1.2, -1.5, -1.7], index=['two', 'four', 'five'])
- frame2['debt'] = val
- Shows
- year state pop debt
- one 2000 Ohio 1.5 NaN
- two 2001 Ohio 1.7 -1.2
- three 2002 Ohio 3.6 NaN
- four 2001 Nevada 2.4 -1.5
- five 2002 Nevada 2.9 -1.7
- six 2003 Nevada 3.2 NaN

DELETING A COLUMN

- The del method can then be used to remove this column:
- del frame2['eastern']
- frame2.columns
- Index(['year', 'state', 'pop', 'debt'], dtype='object')
- The column returned from indexing a DataFrame is a view on the underlying data, not a copy.
- Thus, any in-place modifications to the Series will be reflected in the DataFrame.
- The column can be explicitly copied with the Series's copy method.

DATA FRAME: FILTERING

Any Boolean operator can be used to subset the data:

> greater; >= greater or equal;

<   less;   <= less or equal;

== equal;   != not equal;

DATA FRAMES: SLICING

There are a number of ways to subset the Data Frame:

- one or more columns
- one or more rows
- a subset of rows and columns

Rows and columns can be selected by their position or label

MISSING VALUES

- When summing the data, missing values will be treated as zero
- If all values are missing, the sum will be equal to NaN
- cumsum() and cumprod() methods ignore missing values but preserve them in the resulting arrays
- Missing values in GroupBy method are excluded (just like in R)
- Many descriptive statistics methods have *skipna* option to control if missing data should be excluded . This value is set to *True* by default (unlike R)

PANDAS DATATYPES

- Most of the time, using pandas default int64 and float64 will work.
- Object data type :
  - data type can actually contain multiple different types.
  - For instance, the a column could include integers, floats and strings which collectively are labeled as an object

HANDLING MISSING VALUES

- Python libraries represent missing numbers as nan which is short for "not a number".
- You can detect which cells have missing values, and then count how many there are in each column with the command:
- missing_by_column = (data.isnull().sum())
- print(missing_by_column[missing_by_column > 0)

HANDLING MISSING VALUES:TRAINING AND TEST DATA

- In many cases, you'll have both a training dataset and a test dataset. You will want to drop the same columns in both DataFrames. In that case, you would write
- cols_with_missing = [col for col in original_data.columns
- if original_data[col].isnull().any()]
- reduced_original_data = original_data.drop(cols_with_missing, axis=1)
- reduced_test_data = test_data.drop(cols_with_missing, axis=1)

PYTHON-NUMPY

• Numerical Python (Numpy) has greater role for numerical computing in Python.

• It provides the data structures, algorithms, and library glue needed for most scientific applications involving numerical data in Python.

• It has fast and efficient multidimensional (N-dimensional) array object ndarray

• Functions for performing element-wise computations with arrays or mathematical operations between arrays

• It has tools for reading and writing array-based datasets to disk

SYNTAX

• numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

dtype : Desired data type of array, optional

copy : Optional. By default (true), the object is copied

order : C (row major) or F (column major) or A (any) (default)

subok : By default, returned array forced to be a base class

array. If true, sub-classes passed through

ndmin : Specifies minimum dimensions of resultant array

a = np.array(1,2,3,4) # WRONG a = np.array([1,2,3,4]) # RIGHT

• The function zeros creates an array full of zeros

, • the function ones creates an array full of ones,

• the function empty creates an array whose initial content is random and depends on the state of the memory.

• By default, the dtype of the created array is float64.

• np.ones((2,3,4), dtype=np.int16)

• np.zeros((3,4))

• np.empty((2,3))

UNIVERSAL FUNCTIONS

• NumPy provides familiar mathematical functions such as sin, cos, and exp. In NumPy, these are called “universal functions”(ufunc).

• Within NumPy, these functions operate elementwise on an array, producing an array as output.

BROADCASTING

• Broadcasting refers to the ability of NumPy to treat arrays of different shapes during arithmetic operations. Arithmetic operations on arrays are usually done on corresponding elements.

WHAT IS HTML?

- HTML stands for Hyper Text Markup Language

- HTML is the standard markup language for creating Web pages
- HTML describes the structure of a Web page
- HTML consists of a series of elements
- HTML elements tell the browser how to display the content
- HTML elements label pieces of content such as "this is a heading", "this is a paragraph", "this is a link", etc.
- The `<!DOCTYPE html>` declaration defines that this document is an HTML5 document
- The `<html>` element is the root element of an HTML page
- The `<head>` element contains meta information about the HTML page
- The `<title>` element specifies a title for the HTML page (which is shown in the browser's title bar or in the page's tab)
- The `<body>` element defines the document's body, and is a container for all the visible contents, such as headings, paragraphs, images, hyperlinks, tables, lists, etc.
- The `<h1>` element defines a large heading
- The `<p>` element defines a paragraph

HTML DOCUMENTS

All HTML documents must start with a document type declaration: `<!DOCTYPE html>`.

The HTML document itself begins with `<html>` and ends with `</html>`.

The visible part of the HTML document is between `<body>` and `</body>`.

The `<!DOCTYPE>` declaration represents the document type, and helps browsers to display web pages correctly.

It must only appear once, at the top of the page (before any HTML tags).

The `<!DOCTYPE>` declaration is not case sensitive.

HTML HEADINGS

HTML headings are defined with the `<h1>` to `<h6>` tags.

HTML LINKS

HTML links are defined with the `<a>` tag

HTML IMAGES

HTML images are defined with the `<img>` tag.

The source file (`src`), alternative text (`alt`), `width`, and `height` are provided as attributes

HTML ATTRIBUTES

- All HTML elements can have **attributes**
- Attributes provide **additional information** about elements
- Attributes are always specified in **the start tag**

- Attributes usually come in name/value pairs like: **name="value"**

WHAT IS CSS?

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- CSS saves a lot of work. It can control the layout of multiple web pages all at once
- External stylesheets are stored in CSS files

CSS SYNTAX

The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

CSS SELECTORS

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

This page will explain the most basic CSS selectors.

THE CSS ELEMENT SELECTOR

The element selector selects HTML elements based on the element name.

THE CSS ID SELECTOR

The id selector uses the id attribute of an HTML element to select a specific element.

The id of an element is unique within a page, so the id selector is used to select one unique element!

To select an element with a specific id, write a hash (#) character, followed by the id of the element.

THE CSS CLASS SELECTOR

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

WHAT IS BOOTSTRAP?

- Bootstrap is a free front-end framework for faster and easier web development
- Bootstrap includes HTML and CSS based design templates for typography, forms, buttons, tables, navigation, modals, image carousels and many other, as well as optional JavaScript plugins
- Bootstrap also gives you the ability to easily create responsive designs

BOOTSTRAP 4 JUMBOTRON

A jumbotron indicates a big grey box for calling extra attention to some special content or information.

WHAT IS MACHINE LEARNING

In the real world, we are surrounded by humans who can learn everything from their experiences with their learning capability, and we have computers or machines which work on our instructions. But can a machine also learn from experiences or past data like a human does? So here comes the role of **Machine Learning**.

CLASSIFICATION OF MACHINE LEARNING

At a broad level, machine learning can be classified into three types:

1) SUPERVISED LEARNING

In supervised learning, sample labeled data are provided to the machine learning system for training, and the system then predicts the output based on the training data.

Supervised learning can be grouped further in two categories of algorithms:

- **Classification**
- **Regression**

2) UNSUPERVISED LEARNING

Unsupervised learning is a learning method in which a machine learns without any supervision.

The training is provided to the machine with the set of data that has not been labeled, classified, or categorized, and the algorithm needs to act on that data without any supervision. The goal of unsupervised learning is to restructure the input data into new features or a group of objects with similar patterns.

In unsupervised learning, we don't have a predetermined result. The machine tries to find useful insights from the huge amount of data. It can be further classifieds into two categories of algorithms:

- **Clustering**
- **Association**

3) REINFORCEMENT LEARNING

Reinforcement learning is a feedback-based learning method, in which a learning agent gets a reward for each right action and gets a penalty for each wrong action. The agent learns automatically with these feedbacks and improves its performance. In reinforcement learning, the agent interacts with the environment and explores it. The goal of an agent is to get the most reward points, and hence, it improves its performance.

The robotic dog, which automatically learns the movement of his arms, is an example of Reinforcement learning.

SIGNAL VS. NOISE

• In predictive modeling, you can think of the “signal” as the true underlying pattern that you wish to learn from the data.

• “Noise,” on the other hand, refers to the irrelevant information or randomness in a dataset.

• For example, let’s say you’re modeling height vs. age in children. If you

sample a large portion of the population, you’d find a pretty clear relationship:

• This is the signal.

• However, if you could only sample one local school, the relationship might be muddier. It would be affected by outliers (e.g. kid whose dad is an NBA player) and randomness (e.g. kids who hit puberty at different ages).

NOISE INTERFERES WITH SIGNAL.

• Here’s where machine learning comes in. A well functioning ML algorithm will separate the signal from the noise.

• If the algorithm is too complex or flexible (e.g. it has too many input features or it’s not properly regularized), it can end up “memorizing the noise” instead of finding the signal.

• This overfit model will then make predictions based on that noise. It will perform unusually well on its training data… yet very poorly on new, unseen data.

GOODNESS OF FIT

In statistics, goodness of fit refers to how closely a model’s predicted values match the observed (true) values.A model that has learned the noise instead of the signal is considered “overfit” because it fits the training dataset but has poor fit with new datasets.While the black line fits the data well, the green line is overfit.

OVERFITTING

• Overfitting means that model we trained has trained “too well” and isnow, well, fit too closely to the training dataset.

• This usually happens when the model is too complex (i.e. too manyfeatures/variables compared to the number of observations).

• This model will be very accurate on the training data but will probablybe very not accurate on untrained or new data. It is because thismodel is not generalized (or not AS generalized), meaning you cangeneralize the results and can’t make any inferences on other data,which is, ultimately, what you are trying to do.

• Basically, when this happens, the model learns or describes the“noise” in the training data instead of the actual relationshipsbetween variables in the data. This noise, obviously, isn’t part in ofany new dataset, and cannot be applied to it.

UNDERFITTING

• In contrast to overfitting, when a model is underfitted, it means that the model doesnot fit the training data and therefore misses the trends in the data.

• It also means the model cannot be generalized to new data. This is usually the result ofa very simple model which has not enough predictors/independent variables.

• It could also happen when, for example, we fit a linear model (like linear regression) todata that is not linear. This model will have poor predictive ability (on training data andcan’t be generalized to other data).

• Train/test split and cross validation are two rather important concepts in data scienceand data analysis used as tools to prevent (or at least minimize) overfitting andunderfitting.

• But the tools help to avoid overfitting more than underfitting.

UNDERFITTING

• Underfitting occurs when a model is too simple – informed by too few features or regularized too much – which makes it inflexible in learning from the dataset.

• Simple learners tend to have less variance in their predictions but more bias towards wrong outcomes.

• On the other hand, complex learners tend to have more variance in their predictions.

• Both bias and variance are forms of prediction error in machine learning.

• Typically, we can reduce error from bias but might increase error from variance as a result, or vice versa.

CROSS-VALIDATION

• Cross-validation is a powerful preventative measure against overfitting.

• The idea is clever: Use your initial training data to generate multiple mini train-test splits. Use these splits to tune your model.

K-FOLD CROSS-VALIDATION

Cross-validation allows you to tune hyperparameters with only your original training set. This allows you to keep your test set as a truly unseen dataset for selecting your final model.

CROSS VALIDATION

• Meaning, we split our data into k subsets, and train on k-1 one of those subset. What we do is to hold the last subset for test. We're able to do it for each of the subsets.

CROSS VALIDATION

• There are a bunch of cross validation methods:

• K-FOLDS CROSS VALIDATION

• Leave One Out Cross Validation (LOOCV).In K-Folds Cross Validation we split our data into k different subsets (or folds). We use k-1 subsets to train our data and leave the last subset (or the last fold) as test data. We then average the model against each of the folds and then finalize our model. After that we test it against the test set.

DATA RESCALING

• Data must be prepared before you can build models. The data preparation process can involve three steps: data selection, data preprocessing and data transformation.

• Your preprocessed data may contain attributes with a mixtures of scales for various quantities such as dollars, kilograms and sales volume.

• Many machine learning algorithms perform better when numerical input variables are scaled to a standard range. Scaling the data means it helps to Normalize the data within a particular range.

SCIKIT LEARN METHODS FOR SCALING

• MinMaxScaler, RobustScaler, StandardScaler,and Normalizer are Scikit Learn methods to preprocess data formachine learning.

• Which method you need, if any, depends on your model type and your feature values.

DATA NORMALIZATION

• Normalization refers to rescaling real valued numeric attributes into the range 0 and 1.

• Normalization requires that you know or are able to accurately estimate the minimum and maximum observable values.

• It is useful to scale the input attributes for a model that relies on the magnitude of values, such as distance measures used in k-nearest neighbors and in the preparation of coefficients in regression.

• The example demonstrates data normalization of the Iris flowers dataset.

# Normalize the data attributes for the Iris dataset.

SCIKIT-LEARN METHODS

• Two popular data scaling methods are normalization and standardization.

• The following methods are in sklearn.preprocessing

• MinMaxScalar for Normalisation

• StandardScalar for Standardization

• MinMaxScaler is also known as Normalization and it transform all the values in range between (0 to 1)

x = [(value - min)/(Max- Min)]

• StandardScaler comes under Standardization and its value ranges between (-3 to +3)

z = [(x - x.mean)/Std_deviation]

MINMAXSCALER

• MinMaxScaler transforms each value in the column proportionally

within the range [0,1].

• This must be the first scaler choice to transform a feature, as it will

preserve the shape of the dataset (no distortion).

• MinMaxScaler doesn’t reduce the importance of outliers.

• MinMaxScaler is a good place to start unless you know you want your feature to have a normal distribution or want outliers to have reduced influence.

DATA NORMALIZATION USING MINMAXSCALER

• Fit the scaler using available training data. For normalization, this means the training data will be used to estimate the minimum and maximum observable values. This is done by calling the fit() function.

• Apply the scale to training data. This means you can use the normalized data to train your model. This is done by calling the transform() function.

• Apply the scale to data going forward. This means you can prepare new data in the future on which you want to make predictions.

STANDARDIZATION

• StandardScaler() will transform each value in the column to range about the mean 0 and standard deviation 1, ie, each value will be normalised by subtracting the mean and dividing by standard deviation. • Use StandardScaler if you know the data distribution is normal, ie, the observations fit a Gaussian distribution (bell curve) with a wellbehaved mean and standard deviation. • The ultimate goal to perform standardization is to bring down all the features to a common scale without distorting the differences in the range of the values.

FIT_TRANSFORM()

• fit_transform() is used on the training data so that we can scale the training data and also learn the scaling parameters of that data.

• The fit method calculates the mean and variance of each of the features present in our data and learns them.

• These learned parameters are then used to scale our test data.

• The transform method then transforms all the features using the respective mean and variance. transform()

• Scaling is applied to the test data too and at the same time without any bias with our model.

• The transform method ensures that the test data should be completely new and a surprise set for our model.

• This method ensures that the the same mean and variance as it is calculated from our training data is used to transform our test data.

• Thus, the parameters learned by our model using the training data will help us to transform our test data.

ACCURACY OF MODELS

1. ADD MORE DATA

• Having more data is always a good idea. It allows the "data to tell for itself," instead of relying on assumptions and weak correlations. Presence of more data results in better and accurate models.

• We normally do not get a choice to increase the size of training data. But while working on a company project, more data must be acquired.

• This reduces the pain of working on limited data sets.

2.TREAT MISSING AND OUTLIER VALUES (IMPUTATION)

• The unwanted presence of missing and outlier values in the training data often reduces the accuracy of a model or leads to a biased model. It leads to inaccurate predictions.

• This is because we don’t analyse the behavior and relationship with other variables correctly.

• So, it is important to treat missing and outlier values well.

TREAT MISSING AND OUTLIER VALUES

• Missing: In case of continuous variables, you can impute the missing values with mean, median, mode. For categorical variables, you can treat variables as a separate class. You can also build a model to predict the missing values.

• Outlier: You can delete the observations, perform transformation, binning, Imputation (Same as missing values) or you can also treat outlier values separately.

3. FEATURE ENGINEERING

• This step helps to extract more information from existing data. New information is extracted in terms of new features.

• These features may have a higher ability to explain the variance in the training data. Thus, giving improved model accuracy.

• Feature engineering is highly influenced by hypotheses generation. Good hypothesis result in good features.

• Feature engineering process can be divided into two steps:

A) Feature Transformation B) Feature Creation Feature Engineering Feature transformation:

There are various scenarios where feature transformation is required: A) Changing the scale of a variable from original scale to scale between zero and one. This is known as data normalization.

• Example: If a data set has 1st variable in meter, 2nd in centi-meter and 3rd in kilometer, in such case, before applying any algorithm, we must normalize these variable in same scale. B) Some algorithms works well with normally distributed data. Therefore, we must remove skewness of variable(s). There are methods like log, square root or inverse of the values to remove skewness.

4. FEATURE SELECTION

Feature Selection is a process of finding out the best subset of attributes which better explains the relationship of independent variables with target variable.

• Statistical Parameters: We also consider the p-values, information values and other statistical metrics to select right features.

• PCA: It helps to represent training data into lower dimensional spaces, but still characterize the inherent relationships in the data.

• It is a type of dimensionality reduction technique. There are various methods to reduce the dimensions (features) of training data like factor analysis, low variance, higher correlation, backward/ forward feature selection and others.

5. MULTIPLE ALGORITHMS

• Hitting at the right machine learning algorithm is the ideal approach to achieve higher accuracy. But, it is easier said than done.

• This intuition comes with experience and incessant practice.

• Some algorithms are better suited to a particular type of data sets than others.

• Hence, we should apply all relevant models and check the performance.

• USE CHEAT SHEET

6. ALGORITHM TUNING

• We know that machine learning algorithms are driven by parameters. These parameters majorly influence the outcome of learning process.

• The objective of parameter tuning is to find the optimum value for each parameter to improve the accuracy of the model.

• To tune these parameters, you must have a good understanding of these meaning and their individual impact on model. You can repeat this process with a number of well performing models.
• For example: In random forest, we have various parameters like max_features, number_trees, random_state, oob_score and others. Intuitive optimization of these parameter values will result in better and more accurate models.

7. ENSEMBLE METHODS

• This is the most common approach found majorly in winning solutions of Data science competitions.

• This technique simply combines the result of multiple weak models and produce better results. This can be achieved through many ways:

• Bagging (Bootstrap Aggregating)

• Boosting Ensemble methods

• It is always a better idea to apply ensemble methods to improve the accuracy of your model.

• There are two good reasons for this: a) They are generally more complex than traditional methods. b) The traditional methods give you a good base level from which you can improve and draw from to create your ensembles.

8. CROSS VALIDATION:

• Cross Validation is one of the most important concepts in data modeling.

• It says, try to leave a sample on which you do not train the model and test the model on this sample before finalizing the model.

METRICS FOR EVALUATING PERFORMANCE OF THE MODELS

We can obtain the accuracy score from scikit-learn, which takes as inputs the actual labels and the predicted labels

• from sklearn.metrics import accuracy_score

• accuracy_score(df.actual_label.values, df.predicted_RF.values)

• Shows answer like 0.6705165630156111 Classification Accuracy limitations

• Classification accuracy alone can be misleading if you have an unequal number of observations in each class or if you have more than two classes in your dataset.

2. CONFUSION MATRIX

• A clean and unambiguous way to present the summary of prediction results of a classifier.

• Calculating a confusion matrix can give you a better idea of what your classification model is getting right and what types of errors it is making.

• The number of correct and incorrect predictions are summarized with count values and broken down by each class

• The confusion matrix shows the ways in which your classification model is confused when it makes predictions. Metrics for Evaluating Performance of the Models Process for calculating a confusion Matrix

• You need a test dataset or a validation dataset with expected outcome values.

• Make a prediction for each row in your test dataset.

• From the expected outcomes and predictions count:

• The number of correct predictions for each class.

• The number of incorrect predictions for each class, organized by the class that was predicted.

METRICS FOR EVALUATING PERFORMANCE OF THE MODELS

• These numbers are then organized into a table, or a matrix as follows:

• Expected down the side: Each row of the matrix corresponds to a predicted class.

• Predicted across the top: Each column of the matrix corresponds to an actual class.

• The counts of correct and incorrect classification are then filled into the table.

PROJECTS DONE :

1-1st Glitch Project:
https://visit-.glitch.me/

2- 2nd Glitch Project :
https://shrii-portfolio-.glitch.me/

3- 1st Python Project :
https://colab.research.google.com/drive/1fUHzj2AfoudalZwavQJkCqn1-G7qxCvM?usp=sharing

4-2nd Python Project:
https://colab.research.google.com/drive/1SgOT66URWFEUy4F5WvW1yIYX2K6CNapo?usp=sharing

5- MLProject:
https://pass-fail.glitch.me/