

* 1. **Overview of the Example:**

This example contains the data from the last 14 years(2006/07 to 2018/19) of the English Premire League. Data contains certain features and it is filtered further. Example epl contains SurroundData type object which has the prcoessed data in it. That data is splitted into train and test data which is then predicted using the Logistic Regression. The prediction result and it's analysis is then displayed and saved in the output directory.

![](https://raw.githubusercontent.com/Shibi8/Surround\_AI\_Suqad\_2/master/epl/docs/epl\_flow\_diagram.png)

* 1. **Creating the Project:**

From the project's local folder:

python3 \_\_main\_\_.p

* 1. **Data Selection: Epl.csv** 
     1. **Source:**

The Data utilized in this example has been utilized from the <http://www.football-data.co.uk/englandm.php> website which provides the data for different periods.

* + 1. **Time period:**

The data has been selected for different time periods. We have selected the records from the year 2006/07 till 2018/19.

* 1. **Components** 
     1. **Feeding the Data: Fetch()**

def fetch\_data(surround\_config): dir\_path = surround\_config.get\_path("surround.path\_to\_epl\_data") all\_files = glob.glob(dir\_path + "/\*.csv")

df\_from\_each\_file = (pd.read\_csv(f) for f in all\_files)  
input\_data = pd.concat(df\_from\_each\_file, ignore\_index=True, sort=False)  
return input\_data

In EPL match prediction `fetch\_data` starts by directing the path of `surround\_config` to epl data. path\_to\_epl\_data: ../data/epl\_csv ``Then it grabs all the data from that set path in this case the csv files. Afterwards the csv files are converted into dataframes. In the process of converting these files to dataframes, the index of each file is ignored which in terms gives us theepl\_datawhich is ourinput\_data`.

* 1. **Wrangling ()**

To create different Modulues and frameworks stages( ) are used and it also has multiple classes

It extends the stage class and acts as a skelton for other classess

class WranglingData(Stage):  
   
 def \_\_init\_\_(self):  
 self.processed\_data = pd.DataFrame()

#### let us see different functions involving Wranglingdata(stage)

There are 4 Functions which are used in Wranglingdata(stage) ##### clean\_empty\_data def clean\_empty\_data(self): self.processed\_data.dropna(inplace=True)

This function clean the row containing empty field, that means this function deletes the row in the data folder if it contains any empty field. Remove the row with the missing value of any attributes.

##### **Categorical\_team\_to\_ordinal\_value**

def categorical\_team\_to\_ordinal\_value(self, team):  
 elite\_team = ["""'Man United'""", 'Liverpool', 'Arsenal', 'Chelsea', 'Tottenham']  
 semi\_elite\_team = ["""'Man City'""", 'Everton', """'Aston Villa'""", 'Newcastle', 'Wolves']  
 pro\_team = ["""'Nottingham Forest'""", 'Blackburn', """'Sheffield Wednesday'""", 'Sunderland',  
 """'Leeds United'"""]  
 semi\_pro\_team = ["""'West Brom'""", """'West Ham'""", """'Sheffield United'""", 'Leicester', 'Portsmouth']

This categorical\_team to ordinal value is used tp convert log value in the data tables to integer value and This function assign the rank of each team.

##### **Normalize\_data**

def normalize\_data(self):  
 columns\_to\_standardize = [['HomeTeam', 'AwayTeam', 'HS', 'AS', 'HST', 'AST', 'HF', 'AF', 'HC', 'AC', 'HR', 'AR']]  
  
 for each\_col in columns\_to\_standardize:  
 self.processed\_data[each\_col] = normalize(self.processed\_data[each\_col])

This function normalizes the features on the columns\_to\_normalize. that means in the data obtained for data prediction the values for some of the matches are too high and for some its too low,to make it between (0-1) the data is normailzed. This process can be useful if you plan to use a quadratic form such as the dot-product or any other kernel to quantify the similarity of any pair of samples.

##### **Shift\_column**

def shift\_column(self):  
 temp\_store = self.processed\_data.pop('FTR')  
 self.processed\_data['FTR'] = temp\_store

This function shift column from middle to the last of the dataframe, for the training and testing purpose where 30 of it is testing and 70% is training. The training set contains a known output and the model learns on this data in order to be generalize to other data later on. We have the test dataset (or subset) in order to test our model's prediction on this subset.

* 1. **Modeling and Prediction()**

**Introduction:**

The main purpose of this stage is to perform prediction task on the provided data frames. This stage consists of five different functions.

1. **Initializing Function:**

![Init](https://raw.githubusercontent.com/Shibi8/Surround\_AI\_Suqad\_2/master/Images/init.png)

The def \_\_init\_\_() function initializes the two different variables for the data frames. One variable is for the processed data, which is in the streamline and the other is for the data frame on which the prediction function will be implemented.

1. **Combining Results Function:**

![Combine](https://raw.githubusercontent.com/Shibi8/Surround\_AI\_Suqad\_2/master/Images/combine.png)

This function performs the task of combining multiple testing data subsets of the original dataset and then display the output as the predicted value.

1. **Prediction Report Function:**

![PredictRepo](https://raw.githubusercontent.com/Shibi8/Surround\_AI\_Suqad\_2/master/Images/predict.png)

This function provides performance metrics and other metrics reports for the corresponding prediction models. This function predicts the total number of win for a particular team. It takes into account the total number of home and away wins to predict the result.

Furthermore, the function build a confusion matrix of home win, away wins & draw matches and takes the mean of the home wins in real and predicted results.

1. **Train Test Data Function:**

![TrainTest](https://raw.githubusercontent.com/Shibi8/Surround\_AI\_Suqad\_2/master/Images/TrainTest.png)

The main purpose of this function is to split the datasets into two categories i.e. training and testing datasets into defined ration for further processing. The datasets has been divided as 70% training and 30% testing data. We have utilized logistic regression for our predictive analysis. Here, the test data set is utilized to test the model in consideration to examine its accuracy.

1. **Operate Function:**

![Operate](https://raw.githubusercontent.com/Shibi8/Surround\_AI\_Suqad\_2/master/Images/operate.png)

This function is used to operate processed input data from a datasets onto a resulting output data matrix.

1. **DisplayOutput**

**DataOutput(Stage)** is the final stage in the surround framework. It comes after the stage DataModeling and works as the name suggests. The purpose of this stage is to show the final output of the processed data in the framework. For the example EPL match prediction, the final output of the data can be shown as either of the following \* Graph \* Table

The DataOutput(Stage) can be farther broken down to three different parts.

### Display\_histogram

def display\_histogram(self, histo\_data, output\_dir):  
  
 fig1, ax = plt.subplots(1, 2)  
 histo\_data.FTR.value\_counts().plot("bar", ax=ax[0]).set\_title("Real")  
 histo\_data.Predicted\_FTR.value\_counts().plot("bar", ax=ax[1]).set\_title("Predicted")  
 plt.show()  
  
 fig1.savefig(output\_dir + "/compare\_result.png")  
 plt.close()

From the code we can see that plt is the called function matplotlib.pyplot which is responsible for plotting the histogram. And the histo\_data is the data derived from the previous classes to graphically represent the predection bars. The output we get from display\_histogram is given below.

![](https://raw.githubusercontent.com/Sikbatullah/Surround\_AI\_Suqad\_2/master/epl/output/compare\_result.png)

### Prediction\_report

def prediction\_report(self, real, prediction, output\_dir):  
 h = 0  
 a = 0  
 d = 0  
 for x in prediction:  
 if x == "H":  
 h = h + 1  
 elif x == "A":  
 a = a + 1  
 else:  
 d = d + 1  
 print("Total home win: {0}".format(h))  
 print("Total away win: {0}".format(a))  
 print("Total draw: {0}".format(d))  
  
 labels = ["H", "D", "A"]  
 cnf\_matrix = metrics.confusion\_matrix(real, prediction, labels)  
 print(cnf\_matrix)  
 print("The accuracy of the Logistic regression model is: {0}" .format(metrics.accuracy\_score(real, prediction)))  
  
 df\_confusion = pd.DataFrame(cnf\_matrix, index=["Home Win", "Draw", "Away Win"], columns=["Predicted Home Win", "Predicted Draw", "Predicted Away Win"])  
  
 fig2 = plt.figure(2)  
 sns.heatmap(df\_confusion, annot=True, cbar=False)  
 fig2.savefig(output\_dir + "/confusion\_matrix.png")  
 plt.show()

From the code above we can see that sns is the called function for seaborn which is responsible for plotting the confusion matrix. The confusion matrix compares the real data with the predicted data. Here H represents home win, A represents Away win and D represent the chances of draw between two teams.

![](https://raw.githubusercontent.com/Sikbatullah/Surround\_AI\_Suqad\_2/master/epl/output/confusion\_matrix.png)

### Operate

``` def operate(self, surround\_data, config): output\_dir = config.get\_path("surround.path\_to\_output") surround\_data.result\_data.to\_csv(output\_dir + "/test\_output.csv")

self.display\_histogram(surround\_data.result\_data, output\_dir)  
 self.prediction\_report(surround\_data.result\_data.FTR, surround\_data.result\_data.Predicted\_FTR, output\_dir)  
 print("It's fine up to here.")

``` Operate is responsible changing saving the predictive out test cases to the output folder in csv formats. It is also resposnible for the plotting and saving histograms along with plotting confusion matrix and saving the files.

1. **Features Utilized of Surround:**

### 1. Modular Programming:

Division of a program/class into sub-classes and testing separately makes it easy to read, maintain and rely upon. When a modular system is created, several modules are built separately and more or less independently. The executable application will then be created by putting them together. In other words, Every file, which has the file extension .py and consists of proper Python code is a module. There is no special syntax required to make such a file a module. Usually, modules contain functions or classes, but there can be "plain" statements in them as well. These statements can be used to initialize the module. They are only executed when the module is imported.

### 2. Re-usability of code (Stages)

1. Same code can be used in other code to perform specific functionality as the code is divided into stages.
2. Stage is an implementation of data transformation. Here is the place SurroundData is altered to accomplish the outcome that you need. Each stage is just meant to execute a set of related actions. First stage can be where you get ready information to be processed and last stage can be the place your populate information to be sent back to the client.

### 3. Easier Access to Data (SurroundData)

1. SurroundData is a sharable item between stages that holds vital data for each stage. A stage will read some data from SurroundData, process it, then at that point set back new data that will be utilized by different stage(s). When you expand this class, you can include as many number of variables as you require to enable you to change input data into output data. In any case, there are 4 center factors that are being utilized.

* **stage\_metadata** is information that can be used to identify a stage.
* **execution\_time** is recorded time to complete a process.
* **errors** is information to identify failure of a stage.
* **warnings** is information when transformation is not 100% right.

1. class SVRData(SurroundData):  
     
    def \_\_init\_\_(self):  
    self.dta = pd.DataFrame()  
     
     
    def get\_data(self):  
    self.dta = pd.read\_csv('config.yaml')  
    self.dates = []  
    self.prices = []  
    self.x = [self.prices]

### 4. Surround(): creates Pipeline

1. It is a group of numerous stages or just an initial stage to change raw information into meaningful data. You can set the order of stages directly or by means of a config file. The config file enables you to characterize more than 1 pipeline execution and after that you can switch between them effectively.
2. A file is created with extension .surround to create a project in the pipeline.

* project-info:  
   project-name: svr

### 5. Config.yaml can be configured in the start for the whole project, Global variable, methods

1. Path to data - 'Surround\_AI\_Suqad\_2/svr/data/AAPL.csv'
2. An example from Apple Stock Price Predictions

* class SVRData(SurroundData):
* def **init**(self): self.dta = pd.DataFrame()
* def get\_data(self): self.dta = pd.read\_csv('config.yaml') self.dates = [] self.prices = [] self.x = [self.prices]

config.yaml

output:  
text: Hello World  
  
image: svr  
company: yourcompany  
version: latest  
  
#Details of class inheriting from Wrapper  
wrapper-info: svr.wrapper.PipelineWrapper  
  
surround:  
path: '../data/AAPL.csv'

1. YAML is a human-readable data serialization format that takes concepts from Python, ideas from XML and the data format of electronic mail. The configuration file is written in Python code and saved separately. The config.yaml file can be called/read from within a code.
2. **Drawbacks/ Limitations of Surround:**

#### Importing packages:

When you create a python File with class or function and you need to use it somewhere in another file,then you must important the package that you want to use, when tried with command line using python 3, it throws excess. Necessity of use of operate function in every class is yet to be clarified.

#### Plotting graphs

While plotting the graphs It was not directly popping-up new window rather it was showing error, but for surround data visualization is very important, Its drawback is in that its default style isn't always visually appealing, and it can be complex to make the adjustments according to the data.

#### Conclusion

Surround is framework for machine learning pipelines in python which helps to understand the problem easily, Surround consists of group of stages or a single stage which transforms raw data into meaning full data using layers of filters. Each task is performed in each pipeline which makes the data flow easy, where data can be easily manipulated and accessed around the project, Surround helps in managing error handling as it focus on only a single stage rather than the whole part. It has the flexibility for re-usability of code(stages) as the used code can be reused to perform a different function as the code is divided into stages. There are so many merits of surround as it can used anywhere as it makes the understanding very clear and It divides the program or class into sub-classes and testing separately makes it easy to read, maintain and rely upon. It is very easy to access to data that surround data where it is sharable item between that holds data for each stage