Big mart sales prediction

|  |  |
| --- | --- |
| 5th august | Project title confirmation |
| 12th august | Learning phase   * Machine learning * Python basics |
| 19th august | * Python libraries for machine learning   Numpy Pandas Matplotlib and seaborn Sklearn   * Data collection and processing * Maths |
| 26th august | * Machine learning models   Classification model , Regression model, clustering model, association model |
| 2nd September | * Layouts of the documentation required for the project * Literature survery * Jira GitHub |
| 9th September | * Some concepts cleared in the learning phase * Working on the flow and the methodology of the project and dividing the work in individual accordingly |
| 16th September | Confirmation of the flow making changes in the part given to the individual having a look at past projects seeing there way of dealing the problem and the method they have used for there work |
| 23rd September | Project implementation  Collection and Preparation |
| 30th September | Project implementation  EDA Cleaning and Pre processing |
| 7th October | Project implementation  Model building |
| 14th October | Project implementation  Deployment |
| 21st October | Finalising our work making all the documents ready and preparing for the final presentation |

Problem statement

Nowadays, shopping malls and Big Marts keep track of individual item sales data in order to forecast future client demand and adjust inventory management. In a data warehouse, these data stores hold a significant amount of consumer information and particular item details. By mining the data store from the data warehouse, more anomalies and common patterns can be discovered.

AIM: You have to build a solution that should able to predict the sales of the different stores of Big Mart according to the provided dataset.

Diagram

Description automatically generated

Types of supervised learning

Graphical user interface, text, application, email

Description automatically generated

Work flow

Graphical user interface, application

Description automatically generated with medium confidence

**Random**

**Forest**

**Regressor**

%matplotlib is a [magic function](http://ipython.readthedocs.io/en/stable/interactive/tutorial.html#magics-explained) in IPython. I'll quote the relevant documentation here for you to read for convenience:

IPython has a set of predefined ‘magic functions’ that you can call with a command line style syntax. There are two kinds of magics, line-oriented and cell-oriented. Line magics are prefixed with the % character and work much like OS command-line calls: they get as an argument the rest of the line, where arguments are passed without parentheses or quotes. Lines magics can return results and can be used in the right hand side of an assignment. Cell magics are prefixed with a double %%, and they are functions that get as an argument not only the rest of the line, but also the lines below it in a separate argument.

%matplotlib inline [sets the backend of matplotlib to the 'inline' backend](http://ipython.readthedocs.io/en/stable/interactive/plotting.html):

With this backend, the output of plotting commands is displayed inline within frontends like the Jupyter notebook, directly below the code cell that produced it. The resulting plots will then also be stored in the notebook document.

When using the 'inline' backend, your matplotlib graphs will be included in your notebook, next to the code. It may be worth also reading [*How to make IPython notebook matplotlib plot inline*](https://stackoverflow.com/questions/19410042/how-to-make-ipython-notebook-matplotlib-plot-inline) for reference on how to use it in your code.

[matplotlib.pyplot](https://matplotlib.org/2.0.2/api/pyplot_api.html#module-matplotlib.pyplot) is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc. In

t is not correct to perform standardization before splitting the data. In general, **you should not fit any preprocessing algorithm (PCA, StandardScaler...) on the whole dataset, but only on the training set**, and use the fitted algorithm to transform the test set.

Thus, none of the two experiences you propose are correct. What you should do is:

scaler = StandardScaler().fit(X\_train)

train\_sc = scaler.transform(X\_train)

test\_sc = scaler.transform(X\_test)

It is easy to understand if you think of it this way: the test set is used to get an estimate of the performance of the model on **unseen** data. So you should behave as if you didn't have access to the test set while training the algorithm, and this is also valid for cross validation.

When you fit the standard scaler on the whole dataset, information from the test set is used to normalize the training set. This is a common case of "data leakage", which means that information from the test set is used while training the model. This often results in overestimates of the model's performance.

Note that in scikit-learn you can use [Pipelines](https://scikit-learn.org/stable/modules/generated/sklearn.pipeline.Pipeline.html#sklearn.pipeline.Pipeline) in order to chain the preprocessing steps with the estimator, and use it in the cross validation process. This will ensure that the same steps are repeated for each folds of the cross validation process.

Cross-validation is **a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data**. Use cross-validation to detect overfitting, ie, failing to generalize a pattern.

What is a data leak in machine learning?

Data leakage refers to **a mistake that is made by the creator of a machine learning model in which information about the target variable is leaking into the input of the model during the training of the model**; information that will not be available in the ongoing data that we would like to predict on

RandomForest is a tree-based bootstrapping algorithm that combines a certain number of weak learners (decision trees) to construct a powerful model of prediction. For each person learner, a random set of rows and a few randomly selected variables are used to create a decision tree model. Final prediction may be a function of all the predictions made by the individual learners. In the event of a regression problem, the final prediction may be the mean for all predictions. With this algorithm RMSE:0.54 is reached.

**Data Cleaning**

With this insight, we can go ahead and start cleaning the data. With klib this is as simple as calling *klib.data\_cleaning()*, which performs the following operations:

* **cleaning the column names:**This unifies the column names by formatting them, splitting, among others, CamelCase into camel\_case, removing special characters as well as leading and trailing white-spaces and formatting all column names to *lowercase\_and\_underscore\_separated*. This also checks for and fixes duplicate column names, which you sometimes get when reading data from a file.
* **dropping empty and virtually empty columns:**You can use the parameters *drop\_threshold\_cols* and *drop\_threshold\_rows* to adjust the dropping to your needs. The default is to drop columns and rows with more than 90% of the values missing.
* **removes single valued columns:**As the name states, this removes columns in which each cell contains the same value. This comes in handy when columns such as “year” are included while you’re just looking at a single year. Other examples are “download\_date” or indicator variables which are identical for all entries.
* **drops duplicate rows:**This is a straightforward drop of entirely duplicate rows. If you are dealing with data where duplicates add value, consider setting *drop\_duplicates=False.*
* Lastly, and often times most importantly, especially for **memory reduction** and therefore for speeding up the subsequent steps in your workflow, *klib.data\_cleaning()* also**optimizes the datatypes** as we can see below.