Feature Engineering/Preprocessing

Dealing with Missing Data

Delete: Delete the respective row. We can achieve this using Panda's dropna() function.

Replace with summary: For continuous or quantitative variables, either mean/average or mode or median value of the respective column can be used to replace the missing values. Whereas for categorical or qualitative variables, replacing with the mode (most frequent) summation technique works better. We can achieve this using Panda's fillna() function.

Random replace: We can also replace the missing values with a randomly picked value from the respective column.

Using predictive model: Here you can train a regression model for continuous variables and classification model for categorical variables with the available data and use the model to predict the missing values.

Normalizing Data

- A unit or scale of measurement for different variables varies,
- Therefore, an analysis with the raw measurement could be artificially skewed toward the variables with higher absolute values.
- Bringing all the different types of variable units in the same order of magnitude thus eliminates the potential outlier measurements that would misrepresent the finding.
- Two broadly used methods for rescaling data are normalization and standardization.

Normalization: Normalizing data can be achieved by Min-Max scaling; the formula is given below, which will scale all numeric values in the range 0 to 1. Ensure you remove extreme outliers before applying the above technique as it can skew the normal values in your data to a small interval.

$$X_{\text{normalized}} = \frac{(X - X_{\text{min}})}{(X_{\text{max}} - X_{\text{min}})}$$

Standardization: The standardization technique will transform the variables to have a zero mean and standard deviation of one. The formula for standardization is given below and the outcome is commonly known as z-scores.

$$Z = \frac{(X - \mu)}{\sigma}$$

Where μ is the mean and σ is the standard deviation across the feature.

- Standardization has often been the preferred method for various analyses.
- It tells us where each data point lies within its distribution and a rough indication of outliers.

The aim here is to standardize the range of the variables so that each one of them contributes equally to the analysis. More specifically, the reason why it is critical to perform standardization prior to any step, is that the machine learning model are quite sensitive regarding the variances of the initial variables. That is, if there are large differences between the ranges of initial variables, those variables with larger ranges will dominate over those with small ranges (For example, a variable that ranges between 0 and 100 will dominate over a variable that ranges between 0 and 1), which will lead to biased results. So, transforming the data to comparable scales can prevent this problem.

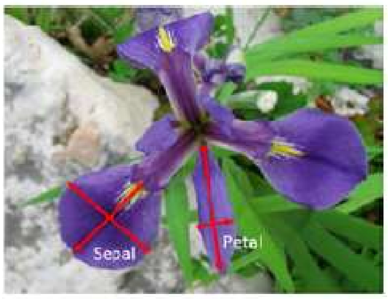
Example 01: Normalization and scaling

```
from sklearn import datasets
import numpy as np
from sklearn import preprocessing
iris = datasets.load_iris()

X = iris.data[:, [2, 3]]
y = iris.target

std_scale = preprocessing.StandardScaler().fit(X)
X_std = std_scale.transform(X)

minmax_scale = preprocessing.MinMaxScaler().fit(X)
X_minmax = minmax_scale.transform(X)
```

Exploratory Data Analysis (EDA)

EDA is all about understanding your data by employing summarizing and visualizing techniques. At a high level the EDA can be performed in two folds, that is, univariate analysis and multivariate analysis.

Let's learn and consider an example dataset to learn practicality. Iris dataset is one of a well-known datasets used extensively in pattern recognition literature. It is hosted at UC Irvine Machine Learning Repository. The dataset contains petal length, petal width, sepal length, and sepal width measurement for three types of iris flowers, that is, setosa, versicolor, and virginica.



Iris versicolor

Figure: Iris flowers

Univariate Analysis

Individual variables are analyzed in isolation to have a better understanding about them. Pandas provide the describe function to create summary statistics in tabular format for all variables. These statistics are very useful for numerical types of variables to understand any quality issues such as missing values and the presence of outliers.

Example 02: Univariate analysis

```
from sklearn import datasets  #importing the database
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

iris = datasets.load_iris()

# Let's convert to dataframe
iris = pd.DataFrame(data= np.c_[iris['data'], iris['target']],
columns= iris['feature_names'] + ['species'])

# replace the values with class labels
iris.species = np.where(iris.species == 0.0, 'setosa', np.where(iris.species==1.0,'versicolor', 'virginica'))

# let's remove spaces from column name
iris.columns = iris.columns.str.replace('','')
iris.describe()

#The columns ‘species’ is categorical, so lets check the frequency distribution for each
#category.

print iris['species'].value_counts()
```

Pandas supports plotting functions to quick visualization on attributes. We can see from the plot that 'species' has 3 categories with 50 records each.

Example 04: Pandas dataframe visualization

```
iris.hist() # plot histogram
plt.suptitle("Histogram", fontsize=16) # use suptitle to add title to all
plt.show()
```

```
iris.boxplot() # plot boxplot
plt.title("Bar Plot", fontsize=16)
plt.show()
```

Multivariate Analysis

In multivariate analysis we try to examine the relationship of all variables with each other. Let's first understand the mean of each feature by species type.

Example 05: Multivariate analysis

```
print(iris.groupby(by = "species").mean())
# plot for mean of each feature for each label class
iris.groupby(by = "species").mean().plot(kind="bar")
plt.title('Class vs Measurements')
plt.ylabel('mean measurement(cm)')
plt.xticks(rotation=0)   # manage the xticks rotation
plt.grid(True)
# Use bbox_to_anchor option to place the legend outside plot area to be tidy
plt.legend(loc="upper left", bbox_to_anchor=(1,1))
```

Correlation Matrix

The correlation function uses Pearson correlation coefficient, which results in a number between -1 to 1. A strong negative relationship is indicated by a coefficient closer to -1and a strong positive correlation is indicated by a coefficient toward 1.

Example 06: Correlation matrix

```
# create correlation matrix
corr = iris.corr()
print(corr)
import statsmodels.api as sm
sm.graphics.plot_corr(corr, xnames=list(corr.columns))
plt.show()
```

Pair Plot

We can understand the relationship attributes by looking at the distribution of the interactions of each pair of attributes. This uses a built-in function to create a matrix of scatter plots of all attributes against all attributes.

#Examplem07: Pair plot

```
from pandas.tools.plotting import scatter_matrix
scatter_matrix(iris, figsize=(10, 10))
# use suptitle to add title to all sublots
plt.suptitle("Pair Plot", fontsize=20)
```