BRAC University
Department of Computer Science and Engineering

CSE427 - Machine Learning

Summer, 2018

Problem Set - 02

Bayesian Linear Regression.

Part - 01: Exploratory Data Analysis

Congratulations! You have been transferred to HYDRA Science division to work under Dr. Zola. Here, you will gather the first-hand experience while working under Dr. Zola. He is considered as the mind behind the HYDRA Science Division.

There is an active war in the shadow, between the HYDRA and the S.H.I.E.L.D. Your assignment here is to work with a secret dataset that contains the students' information from one of the secret Academy of S.H.I.E.L.D, where young agents are trained. You will study the data and make a prediction of the results of the students.

Now, make a new notebook in the Jupyter notebook and follow the steps. We assume that you know the basics of the python as you have successfully done the problem set 1.

Phase 00: Setting the environment

First, we will import the necessary libraries and modules so that we can avoid unexpected errors later.

```
# Pandas and numpy for data manipulation
import os
import pandas as pd
import numpy as np
np.random.seed(42)
```

```
# Matplotlib and seaborn for plotting
import matplotlib.pyplot as plt
%matplotlib inline

import matplotlib
matplotlib.rcParams['font.size'] = 16
matplotlib.rcParams['figure.figsize'] = (9, 9)
```

```
import seaborn as sns

from IPython.core.pylabtools import figsize

# Scipy helper functions
from scipy.stats import percentileofscore
from scipy import stats
```

```python
# Standard ML Models for comparison
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import ElasticNet
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.svm import SVR

# Splitting data into training/testing
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

# Metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error,
                                    median_absolute_error

# Distributions
import scipy
```

```python
# PyMC3 for Bayesian Inference
import pymc3 as pm
```

These libraries are built-in in anaconda. You should face no error while importing these libraries. However, the *pymc*3 package doesn't come pre-installed. To install it, follow these steps.

1. Open your anaconda navigator
2. Go to *environments* form the right navigator bar
3. select the *Not installed* packages from the drop down menu
4. Scroll down till you find the desired package
5. Select it and install it
6. Restart the navigator

Phase 01: Getting the Data

The data is kept in a high-security vault in a secret HYDRA base, you will find a way to download the data from here. You will find two datasets here. Download both datasets to your local repository where you have saved the Jupyter notebook.

Phase 02: Studying the Data

For now, we will study the *student−mat* dataset. Let's see what's inside the *student−mat* dataset. Simply put the below codes in your notebook.

```python
# Read dataset
def load_data():
    csv_path = os.path.join("student-mat.csv")
    return pd.read_csv(csv_path)
```

```
df = load_data()

# Filter out grades that were 0
df = df[~df['Grade'].isin([0, 1])]

df.head()
```

Question 01: What is the *dimension* of the *student − mat* dataset?(Hint: try to use your knowledge from the previous problem set)

The primary variable of the dataset is the Grades of the students. Let's look at the distribution of the Grades to check for skew. Use the codes below.

```
import matplotlib.pyplot as plt
# Histogram of grades
plt.hist(df['Grade'], bins = 14)
plt.xlabel('Grade')
plt.ylabel('Count')
plt.title('Distribution of Final Grades')
plt.show()
```

Change the *bins* as per your preferences and see how the histogram changes. Now, let's see the bar plot of the Grades.

```
# Bar plot of grades
plt.bar(df['Grade'].value_counts().index,
        df['Grade'].value_counts().values,
         fill = 'navy', edgecolor = 'k', width = 1)
plt.xlabel('Grade');
plt.ylabel('Count');
plt.title('Distribution of Final Grades');
plt.xticks(list(range(5, 20)));
```

Question 02: Are the grades normally distributed? If yes, at which point?

As we have seen that the overall grades do not have a noticeable skew, it is possible that students from certain categories will have skewed grades. To find out if the categorical variables have any effect on the grades, we will make density plots of the grade distribution coloured by the values of the categorical variable.

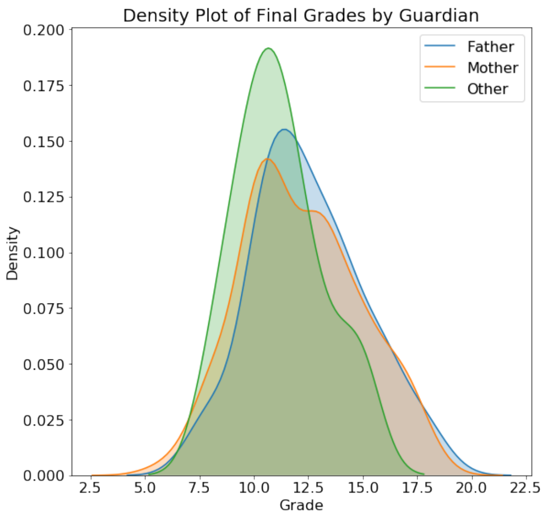
```
# Grade distribution by address
sns.kdeplot(df.ix[df['address'] == 'U', 'Grade'], label = 'Urban',
                                        shade = True)
sns.kdeplot(df.ix[df['address'] == 'R', 'Grade'], label = 'Rural',
                                        shade = True)
plt.xlabel('Grade');
plt.ylabel('Density');
plt.title('Density Plot of Final Grades by Location');
```

Don't worry if you get a *DeprecationWarning* from the python kernel. Just avoid it.

Question 03: **Write a similar block of codes to plot the distribution of the grades by,**

1. **Guardian**
2. **Internet Access**
3. **Schools**

The density plot of final grade by guardian will look something like this.



As the actual values on a density plot are difficult to interpret we can use the shapes of the plots for comparisons. The *grade_guardian* plot shows, the guardians have no substantial impact on the students' grades.

Question 03: **Do the other categorical variables have any impact on the grades of the students? Answer from the graphs that you have plotted.**

Grade Percentiles

Let's calculate the percentile of the grades.

```
# Calculate percentile for grades
df['percentile'] = df['Grade'].apply(lambda x: percentileofscore(df['
                                                Grade'], x))
```

```
# Plot percentiles for grades
plt.figure(figsize = (8, 6))
plt.plot(df['Grade'], df['percentile'], 'o')
plt.xticks(range(0, 20, 2), range(0, 20, 2))
plt.xlabel('Score');
plt.ylabel('Percentile');
plt.title('Grade Percentiles');
```

Question 04: **What is the 50^{th} percentile score? Answer from the percentile graph.**

Feature Selection

We don't expect every variable to be related to the final grade, so we need to perform a feature selection (also called the *dimensionality reduction*) to choose only the "*relevant*" variables. As we will be doing linear modelling, we will use Correlation Coefficient to determine the most useful variables for predicting a grade. This is a value between $+1$ and -1 that measures the direction and strength of a linear relationship of two variable.

To select fewer variables, we have to select those that have the greatest correlation (either positive or negative). Using pandas we can do that very easily.

```
# Find correlations and sort
df.corr()['Grade'].sort_values()
```

From the list, we will ignore the $G1$, $G2$ *and Grade* for now.

Question 04: Write down the list of the correlated variables and underline the values those have most impact on the grades (You can ignore the variables mentioned before).

Correlation can only be calculated between numerical variables. To determine the relationship between the categorical variables and grade, we will use the One-hot encoding. This is a standard step in machine learning pipelines. Using pandas you can do it very easily.

```
# Select only categorical variables
category_df = df.select_dtypes(include = ['object'])
# One hot encode the variables
dummy_df = pd.get_dummies(category_df)
# Put the grade back in the dataframe
dummy_df['Grade'] = df['Grade']
dummy_df.head()
```

Now, if you see the sample of the dummy data, you will notice that we have managed to convert the object data into numerical value. Let's calculate the correlation with the new data.

```
# Correlations in one-hot encoded dataframe
dummy_df.corr()['Grade'].sort_values()
```

This shows the relation between the categorical variables and the grades.

Separating the training data and test data:

Before implementing our main machine learning algorithm, we will need a set of training data and a set of test data. A training data set represents all the characteristics of the actual data. We will use this dataset to train our machine learning model and then we will test our model with the test dataset before deploying to the real world.

We will separate the actual data into two smaller datasets. One will be training set and another will be test set. Before doing this, first we will select 5 most correlated variables with the final grade. Then, we will split the main dataset into two separate datasets, of which one will be a training set, which will contain 25% of the actual data and the test dataset will contain 75% of the actual data.

```
# Takes in a dataframe, finds the most correlated variables with the
# grade and returns training and testing datasets
def format_data(df):
    # Targets are final grade of student
    labels = df['Grade']

    # Drop the school and the grades from features
    df = df.drop(['school', 'G1', 'G2', 'percentile'], axis=1)

    # One-Hot Encoding of Categorical Variables
    df = pd.get_dummies(df)

    # Find correlations with the Grade
    most_correlated = df.corr().abs()['Grade'].sort_values(ascending=
                                          False)

    # Maintain the top 5 most correlation features with Grade
    most_correlated = most_correlated[:8]

    df = df.ix[:, most_correlated.index]
    df = df.drop('higher_no', axis=1)

    # Split into training/testing sets with 25% split
    X_train, X_test, y_train, y_test = train_test_split(df, labels,
                                          test_size = 0.25, random_state=
                                          42)

    return X_train, X_test, y_train, y_test
```

Now that, we have separated the training and test data set, let's look into the training dataset.

```
X_train, X_test, y_train, y_test = format_data(df)
X_train.head()
```

This will show you what's inside the training dataset.

Question 05: What are the dimensions of the training and the test datasets?

Let's rename the variables' name in the training and test dataset.

```
# Rename variables in train and teste
X_train = X_train.rename(columns={'higher_yes': 'higher_edu',
                                  'Medu': 'mother_edu',
                                  'Fedu': 'father_edu'})

X_test = X_test.rename(columns={'higher_yes': 'higher_edu',
                                  'Medu': 'mother_edu',
                                  'Fedu': 'father_edu'})
```

Bonus Question 01: Check if the names of the variables have been changed or not. If yes, then write the names of the changed variables.

So far, we have split the original data into training and test datasets. To know more about these datasets we can use the Pair Plot.

```
# Calculate correlation coefficient
def corrfunc(x, y, **kws):
    r, _ = stats.pearsonr(x, y)
    ax = plt.gca()
    ax.annotate("r = {:.2f}".format(r),
                xy=(.1, .6), xycoords=ax.transAxes,
               size = 24)

cmap = sns.cubehelix_palette(light=1, dark = 0.1,
                             hue = 0.5, as_cmap=True)

sns.set_context(font_scale=2)

# Pair grid set up
g = sns.PairGrid(X_train)

# Scatter plot on the upper triangle
g.map_upper(plt.scatter, s=10, color = 'red')

# Distribution on the diagonal
g.map_diag(sns.distplot, kde=False, color = 'red')

# Density Plot and Correlation coefficients on the lower triangle
g.map_lower(sns.kdeplot, cmap = cmap)
g.map_lower(corrfunc);
```

There is a lot of information encoded in this plot. The scatterplots on the upper triangle shows every variable plotted against one another. Most of the variables are discrete integers. On the diagonal we have histograms, showing all the single variables. the lower triangle has both 2-D density plots and the correlation coefficient between variables.

To interpret the plot, we can select a variable and look at the row and column to find the relationships with all the other variables. For example, the first row shows the scatterplots of Grade (which is out primary variable) with the other variables. The first column shows the correlation coefficient between the Grade and the other variables.

Bonus Question 02: Which variable has the greatest correlation with the final grade in terms of absolute magnitude.

We can also make distribution plots of each variable, colouring the plot by if the grade is above the median score of 12.

```
# Create relation to the median grade column
X_plot = X_train.copy()
X_plot['relation_median'] = (X_plot['Grade'] >= 12)
X_plot['relation_median'] = X_plot['relation_median'].replace({True: '
                                   above', False: 'below'})
X_plot = X_plot.drop('Grade', axis=1)
```

Now, if you make the variables' distribution by relation to median, we will see-

```
plt.figure(figsize=(12, 12))
# Plot the distribution of each variable colored
# by the relation to the median grade
for i, col in enumerate(X_plot.columns[:-1]):
    plt.subplot(3, 2, i + 1)
    subset_above = X_plot[X_plot['relation_median'] == 'above']
    subset_below = X_plot[X_plot['relation_median'] == 'below']
    sns.kdeplot(subset_above[col], label = 'Above Median', color = '
                                          green')
    sns.kdeplot(subset_below[col], label = 'Below Median', color = 'red
                                          ')
    plt.legend(); plt.title('Distribution of %s' % col)

plt.tight_layout()
```

Now, try to understand the relation of the grade with the other variables from the plots.

———————— end of part-01 ————————

In part-02, we will test a few built-in machine learning algorithms to test out dataset. Then we will start implementing the *Bayesian Linear Regression*