Nicolas Thomazo

Data Analysis with Python

Introduction

In this document, I discuss some data analysis concepts with python.

Packages

- Scientifics computing libraries
  Pandas (Data structures), numpy (arrays & matrices), SciPy(integrals,optimization)

- Visualisation
  Matplotlib (plot & graphs), Seaborn (plots : heat maps ,time series)

- Algorithmic
  Scikit-learn (machine learning : regression, classification…), Statsmodels (statistics test, explore data)

Data pre-processing/ Data cleaning

1. Handle missing values

If we don't want to lose data, we can replace it. For example, average, frequency or similar datapoints. Otherwise we can drop it or leave it as missing data.

Dataframe.dropna(axis = 0, inplace = True) -> drop missing values

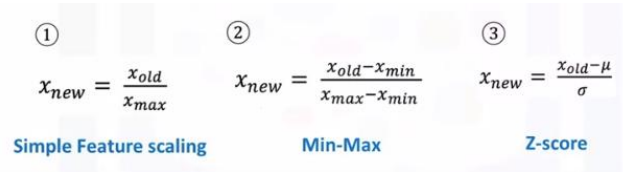
Dataframe.replace (missing_value, new_value) -> replace missing values

2. Data formatting

It means bringing data into a common standard of expression allows users to make meaningful comparison. For example, datatype is object instead of integer, or for one city there are differences way of wright it.

3. Data normalization (cantering /scaling)

It means uniform the features values with different range. If we have two values with different range for example age and income. As the range of income is larger than age, it will influence the result more. To prevent that we normalized them between 0 and 1. Now they will have the same influence on the model.



4. Data binning

Binning : grouping of values into “bins”. For example, if we have a price that goes from 5000 to 45000, we can categorize it between 3 bin “low”, “medium”, “high”

We use the numpy function to get 4 equals intervals. Then we use the pandas function cut to segment and sorted the data into bins.

```
bins = np.linspace(min(df["price"]), max(df["price"]), 4)
group_names = [“Low”, “Medium”, “High”]
df[“price-binned”] = pd.cut(df[“price”], bins, labels=group_names, include_lowest=True )
```

5. Categorical values to numeric values

Most statistical models cannot take in the objects/string as input. We add variables for each unique category.

In pandas we use get_dummies() method to convert categorical variables to dummy variables (0 or 1).

Exploratory Data Analysis

By using descriptive statistics, correlation… in order to summarize the main characteristics of the data and get a better understand the dataset.

1. Descriptive statistics

By using **df.describe()** in a dataframe we have all the statistics for each column. The **value_count()** method to summarize categorical data. **boxplot** method shows a good representation of the data.

2. GroupBy in python

It can be applied on categorical variables only. We can use the pandas method **groupby**. We can use then the **pivot()** method to display one variable along the columns and the other along the rows.

3. correlation

Measures to what extent different variables are interdependent. We use the Pearson correlation method to measure the strength of correlation between two features.

Nicolas Thomazo

Correlation coefficient : close to +1 -> large positive relationship
close to -1 -> large negative relationship
close to 0 -> no relationship.

P-value : value < 0.001 -> String certainty in the result
value < 0.05 -> moderate certainty
value > 0.1 -> no certainty

With Scipy we can measure it easily, for example

```
pearson_coef, p_value = stats.pearsonr(df['horsepower'], df['price'])
```

4. Analysis of variance

It is a statistical comparison of groups. We can use the ANOVA method. It gives us the correlation between different groups of a categorical variable. It returns two values

- F-test score : variation between sample group means divided by variation
- P-value : confidence degree

With Scipy we can measure it, for example

```
df_anova=df[["make", "price"]]
grouped_anova=df_anova.groupby(["make"])

anova_results_l=stats.f_oneway(grouped_anova.get_group("honda")["price"], grouped_anova.get_group("subaru")["price"])
```

Model Development

1. Linear Regression

Understand the relationship between variables. The predictor and the target.

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
X = df[['highway-mpg']]
Y = df['price']
lm.fit(X, Y)
Yhat=lm.predict(X)
```

2. Model evaluation (distplot, regplot, residplot)

We can use the regression plot, that show us a combination of the scatterplot(where each point is represented) and the fitted linear regression line.From the seaborn libraries we can use regplot

```
sns.regplot(x="highway-mpg", y="price", data=df)
plt.ylim(0,)
```

Distribution plot compare the predicted values with the actual values, we can have a histogram or just a curve.

3. Polynomial Regression and pipelines (np.polyfit, scikit-learn Polynom)

Useful for describing curvilinear relationship.

```
f=np.polyfit(x,y,3)

p=np.poly1d(f)
```

We can use pipeline from the scikit leanr libiraires.

4. Measures to determine how good the model fits on the dataset (mean_squared_error, score)

Mean Squarred Error (MSE)
We find the difference between the actual value and the predicted value. To make the value positive we squared the error. Then we calcul the average of all the errors.

R-Squared(R^2)
It is a measure of how close the data is to the fitted regression line, it's a percentage of variation.

$$R^2=\left(1-\frac{\text{MSE of regression line}}{\text{MSE of the average of the data}}\right)$$

It is the percent of variance explained by the model.

Overfitting, Underfitting and model selection

Underfitting : model to simple to predict the data

Overfitting : model is to accurate for the training set and will poorly predict new data

Ridge regression prevent overfitting. It controls the magnitude of the polynomial coefficient by introducing alpha. If alpha is to large, we will have underfitting, if alpha is to small we will have overfitting.

Useful function

Pandas :

- Dataframe.dropna, remove missing values
- Dataframe.replace, Values of the DataFrame are replaced with other values dynamically
- Bin values into discrete intervals. `pd.cut(np.array([1, 7, 5, 4, 6, 3]), 3)` Will make 3 bins of equi depth and return for each value his bin
  ```
  ...
  [(0.994, 3.0], (5.0, 7.0], (3.0, 5.0], (3.0, 5.0],
  ```
- Convert categorical variable into dummy/indicator variables. `pd.Series(list('abca'))` will return
  ```
  >>> pd.get_dummies(s)
     a  b  c
  0  1  0  0
  1  0  1  0
  2  0  0  1
  3  1  0  0
  ```
- Dataframe.groupby, combination of splitting the object, applying a function, and combining the results. For example

Nicolas Thomazo

```
>>> df
   Animal  Max Speed
0  Falcon      380.0
1  Falcon      370.0
2  Parrot       24.0
3  Parrot       26.0
```

```
>>> df.groupby(['Animal']).mean()
        Max Speed
Animal
Falcon      375.0
Parrot       25.0
```

Numpy :

- Polyfit, give it two arrays (x and y) and a degree and returns the value of the polynomial that fit the best the values x and y
- Poly1D, give it one array and return the polynomial
- Linspace, Return evenly spaced numbers over a specified interval. For example

```
>>> np.linspace(2.0, 3.0, num=5)
array([2.  , 2.25, 2.5 , 2.75, 3.  ])
```

Scipy :

- Stats.pearson :explain above
- Anova : explain above

Seaborn:

- Regplot, Plot the relationship between two variables in a DataFrame (points + linear regression model)
- Residplot, can be a useful tool for checking whether the simple regression model is appropriate for a dataset. It fits and removes a simple linear regression and then plots the residual values for each observation. Ideally, these values should be randomly scattered around y = 0: If there is structure in the residuals, it suggests that simple linear regression is not appropriate:
- Distplot, This function combines the matplotlib hist function (with automatic calculation of a good default bin size) with the seaborn kdeplot()

Sklearn :

- StandardScaler() : centering and scaling each variable (z = (x -u)/s). We need to use the fit() and then the transform() method
- PolynomialFeatures() : generate new feature matrix consisting of all polynomial combination (with the degree), [a, b], the degree-2 polynomial features are [1, a, b, a^2, ab, b^2]
- Sequentially apply a list of transforms and a final estimator.Intermediate steps of pipeline must implement fit and transformmethods and the final estimator only needs to implement fit.
- RidgeModel : It prevent overfitting. Ridge regression controls the magnitude of the polynomial coeeficient by introducing the parameter alpha. It is the parameter we select before fitting or training the model. The more alpha is close to 0, the more we will have overfitting