REPORT on Introduction to Theoretical and Practical Data Science

11911819 缪方然 11911915 曹书伟

1.Data exploration

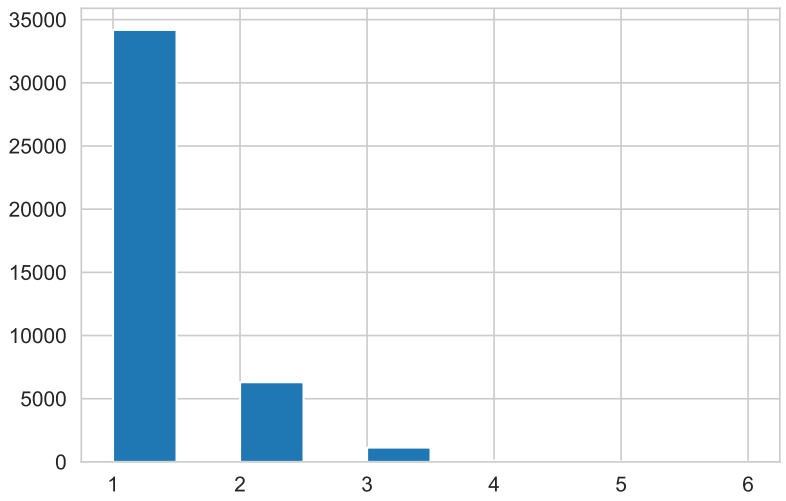
data statistics

	pm2.5	DEWP	TEMP	PRES	Iws	Ir	Is
mean	98.613	1.817	12.449	1016.448	23.889	0.053	0.195
median	72	2	14	1016	5.37	0	0
mode	16	18	24	1014	0.89	0	0
maximum	994	28	42	1046	585.6	27	36
minimum	0	-40	-19	991	0.45	0	0

data visualization

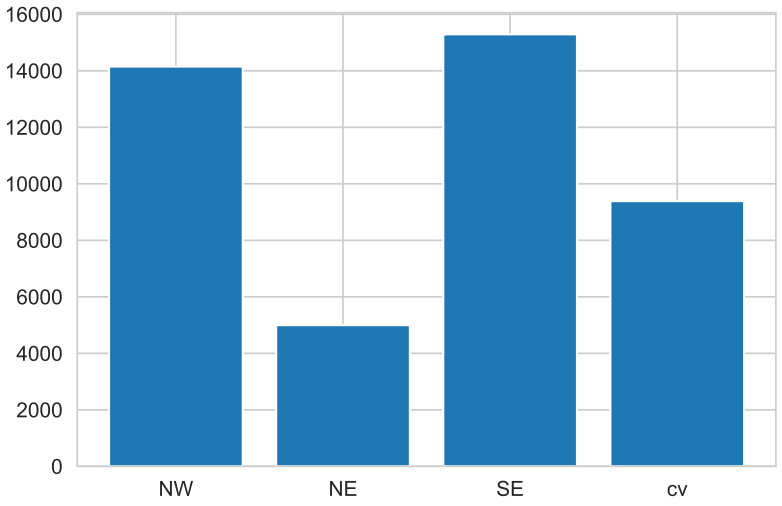
Here we draw some diagrams to show the insight of the data.

- **Hist diagram of the to-predict value--pm2.5**



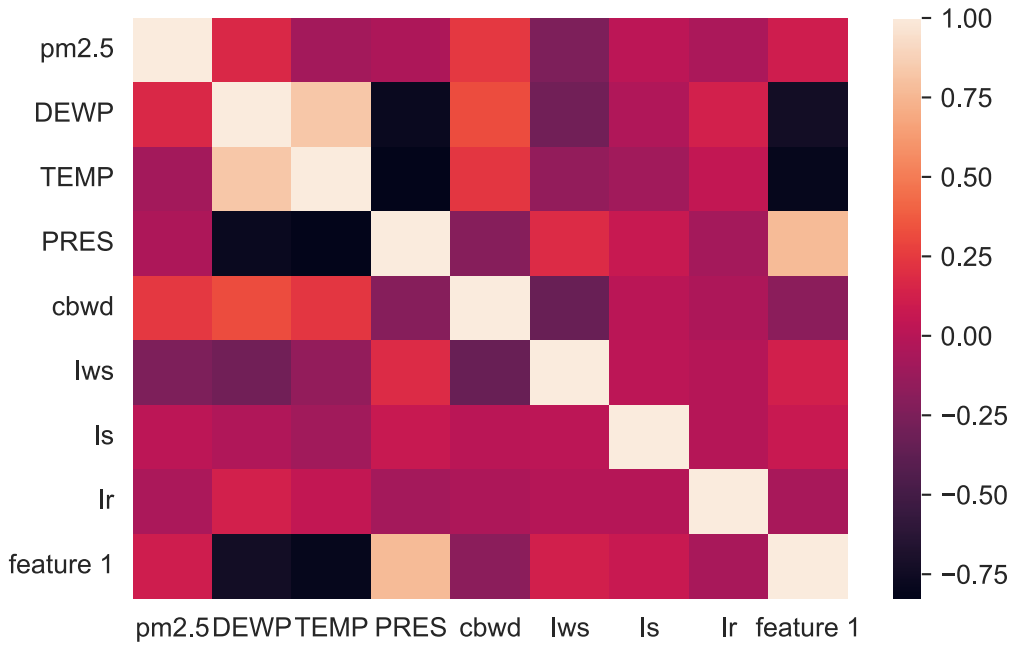
The diagram shows that the value of pm2.5 are mainly in the range of 0 to 165. Although pm2.5 above 500 is merely extinct in the diagram, there are still few of them. (About 20 and even less)

- **Bar diagram of cbwd**



The diagram shows that the southeastern and northwestern wind often blow in Beijing, and it hardly appears southwestern wind.

- **Heatmap**



The diagram shows the correlation between different features. It shows that `PRES` has strong relationship with both `DEWP` and `TEMP`. Meanwhile, `DEWP` and `TEMP` also has strong relationship.

2.Data preprocessing

dataset partition

training dataset and test dataset

We split the `PRSA_data.csv` into training dataset and test dataset

According to the requirements of the problem, one day is selected every seven days as the test set, and the other data are put into the training set.

cross validation

We also do the cross validation for dataset partition to improve the robustness of our model.

detecting missing values

- **Detect**
  We use `np.where` and `np.isnan` to find and locate the NAN data. Then generate the `nan_index` and `non_nan_index` to help us. The related code is in `data_preprocess.py->detect_missiing_data .`

- **Fill**
  We use several methods to fill the NAN data. We use 0, average, mode and median to fill the NAN. Since it will appear large amounts of times, it is, generally, not a proper way.
  Then we use KNN to fill. We find 5 nearest neighbors for a certain row which contains a missing value. And we take the average of the 5 nearest neighbors' values of pm2.5 to fill the NA.
  Last we use interpolation to fill it. The most natural idea is that the value of pm2.5 is continuous about time. Thus, we use the interpolation of pm2.5 versus time.

data conversion

There is only one column need to be converted in the raw data--'cbwd'. First use the function `set` to find the number of unique values. As a result, there are 4-- "NW", "NE", "SE", "cv". Then use `dict` to store a mapping relation, i.e. "NW" maps to 1, "NE" maps to 2, "SE" maps to 3, "cv" maps to 4. Finally, replace in the original data.

data normalization

We use `sklearn.preprocessing.StandardScaler` to normalize data. Note that there are still some features that we don't want to normalize, e.g. "cbwd". Since this kind of data is just a representation of the direction of the wind, and it does not necessarily follow the Gauss Distribution.

data imputation

There are several ways to impute data.

We use `mean value` , `median value` , `mode value` and `0` to impute the missing data in *pm2.5*. We also use method of `KNN` and `interpolate` to impute.

Generally, the performance of first four methods are not so good as the latter two ones.

3.Model construction

In machine learning, we use several models to train our data to do regression and classification.

Regression

The regression models are stored in `regression_models.py` .

- **linear regression model** Implemented by `sklearn.linear_model.LinearRegression` . The related code can be found in `regression_models.py->ordinary_regression` .
- **ridge regression** Implemented by `sklearn.linear_model.Ridge` . The related code can be found in `regression_models.py->ridge_regression` .
- **LASSO regression model** Implemented by `sklearn.linear_model.Lasso` . The related code can be found in `regression_models.py->LASSO_regression` .
- **random forest regressor model** Implemented by `sklearn.ensemble.RandomForestRegressor` . The related code can be found in `regression_models.py->random_forest_regressor` .
- **extra trees regressor** Implemented by `sklearn.ensemble.ExtraTreesRegressor` . The related code can be found in `regression_models.py->extra_trees_regressor` .
- **gradient boosting regressor** Implemented by `sklearn.ensemble.GradientBoostingRegressor` . The related code can be found in `regression_models.py->gradient_boosting_regressor` .
- **support vector regressor** Implemented by `sklearn.svm.svr` . The related code can be found in `regression_models.py->svr` .
- **MLP regressor** Implemented by `sklearn.MLPRegressor` . The related code can be found in `regression_models.py->mlp_regressor` .

Classification

The classification models are stored in `classification_models.py` .

- **KNN** Implemented by `sklearn.neighbors.KNeighborsClassifier` . The related code can be found in `classification_models.py->KNN_classification` .
- **SVM** Implemented by `sklearn.svm.svc` . The related code can be found in `classification_models.py->SVM_classification` .
- **Decision Tree** Implemented by `sklearn.tree.DecisionTreeClassifier` . The related code can be found in `classification_models.py->decision_tree_classification` .
- **Logistic Regression** Implemented by `sklearn.linear_model.LogisticRegression` . The related code can be found in `classification_models.py->decision_tree_classification` .
- **LDA** Implemented by `sklearn.discriminant_analysis.LinearDiscriminantAnalysis` . The related code can be found in `classification_models.py->decision_tree_classification` .
- **MLP classification** Implemented by `sklearn.neural_network.MLPClassifier` . The related code can be found in `classification_models.py->decision_tree_classification` .

4.Feature selection and Model selection and Feature creation

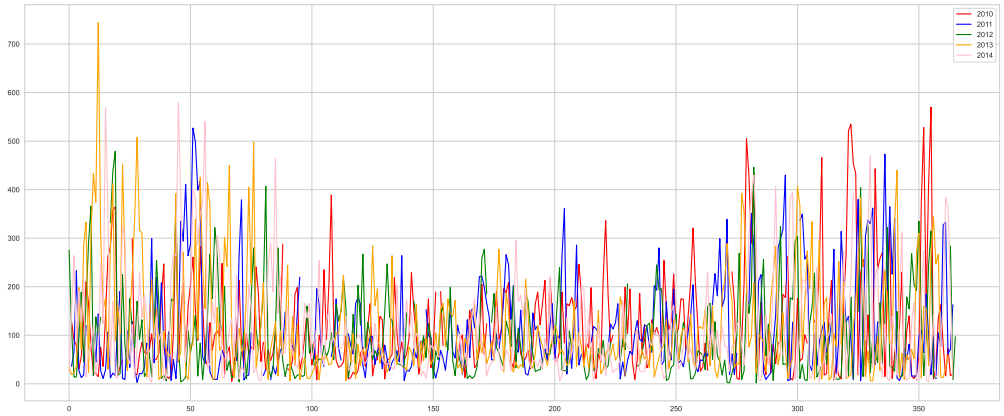
Feature selection

In this part, we use several ways to select features.

- **AIC and BIC**
  We use forward AIC and BIC to select the features step by step. Since scikit-learn doesn't provide the related API, we implemented this part by ourselves.
  Both methods show that `DEWP` , `TEMP` , `PRES` , `cbwd` , `Iws` are the selected feature.
- **LASSO**
  We use decreasing α to select features. The result shows that `Iws` , `DEWP` , `PRES` , `TEMP` , `Ir` , `cbwd` are selected features.
- **Random Forest**
  Random forest can calculate the feature importance. And we used the built-in API in this package. The result shows that `DEWP` , `TEMP` , `Iws` , `PRES` , `cbwd` are important.
- **Mutual Information**
  We use the mutual information to select the features. The result shows that `DEWP` , `Iws` , `cbwd` , `PRES` , `TEMP` are the top 5 important ones.
- **Conclusion**
  We can find out that `DEWP` , `Iws` , `cbwd` , `PRES` , `TEMP` are important features. And `Ir` , `Is` are not so important features.

Feature creation

- **Feature 1**
  By data analysis, the value of pm2.5 is largely effected by the season. We draw the value of pm2.5 at 12a.m. every day for each year and find out that it is extremely high in winter and relatively low in summer.



  Hence, we add our first feature. Since it's high in winter and low in summer, we assign 4 for winter and 1 for summer. With more specific analysis, we find the pollution in fall seems worse than that of spring. Thus, we assign 3 for fall and 2 for spring.

Model selection

5.Model evaluation

R2 score

F1 score

cross validation

6.Conclusion