Red Wine Quality

Reported by: Panfeng Jiang & Fei Pang

Import Libraries & Data loading

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
from termcolor import colored
from imblearn.over_sampling import SMOTE
import matplotlib.patches as patches
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, LSTM, Conv1D, Flatten, BatchNormalization
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import mean_absolute_error
from tensorflow.keras.layers import Conv1D, Flatten
from tensorflow.keras.regularizers import l2
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping
```

Fixed acidity: Content of non-volatile acids in wine.
Volatile acidity: Content of volatile acids in wine.
Citric acid: Content of citric acid.
Residual sugar: Amount of residual sugar in wine.
Chlorides: Content of chlorides in wine.
Free sulfur dioxide: Content of free sulfur dioxide in wine.
Total sulfur dioxide: Content of total sulfur dioxide in wine.
Density: Density of the wine.
pH: pH level of the wine.
Sulphates: Content of sulphates in wine.
Alcohol: Alcohol content of the wine.

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH	sul
0	9.7	0.690	0.32	2.5	0.088	22.0	91.0	0.99790	3.29	
1	6.6	0.580	0.02	2.4	0.069	19.0	40.0	0.99387	3.38	
2	9.2	0.755	0.18	2.2	0.148	10.0	103.0	0.99690	2.87	
3	9.0	0.785	0.24	1.7	0.078	10.0	21.0	0.99692	3.29	
4	10.6	0.360	0.57	2.3	0.087	6.0	20.0	0.99676	3.14	
5	12.0	0.450	0.55	2.0	0.073	25.0	49.0	0.99970	3.10	
6	6.0	0.500	0.00	1.4	0.057	15.0	26.0	0.99448	3.36	
7	6.2	0.560	0.09	1.7	0.053	24.0	32.0	0.99402	3.54	
8	7.8	0.560	0.19	2.0	0.081	17.0	108.0	0.99620	3.32	
9	6.6	0.500	0.01	1.5	0.060	17.0	26.0	0.99520	3.40	

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 12 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed.acidity         1000 non-null   float64
 1   volatile.acidity      1000 non-null   float64
 2   citric.acid           1000 non-null   float64
 3   residual.sugar        1000 non-null   float64
 4   chlorides             1000 non-null   float64
 5   free.sulfur.dioxide   1000 non-null   float64
 6   total.sulfur.dioxide  1000 non-null   float64
 7   density               1000 non-null   float64
 8   pH                    1000 non-null   float64
 9   sulphates             1000 non-null   float64
 10  alcohol               1000 non-null   float64
 11  quality               1000 non-null   int64
dtypes: float64(11), int64(1)
memory usage: 93.9 KB
```

Red Wine Quality Dataset: 11 input variable, 1 output variable.

Data Info & cleaning

```
data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
fixed.acidity	1000.0	8.301300	1.713092	4.70000	7.100000	7.90000	9.200000	15.9000
volatile.acidity	1000.0	0.523645	0.172574	0.12000	0.395000	0.51000	0.630000	1.5800
citric.acid	1000.0	0.268310	0.194616	0.00000	0.090000	0.25000	0.420000	1.0000
residual.sugar	1000.0	2.552300	1.429940	0.90000	1.900000	2.20000	2.600000	15.5000
chlorides	1000.0	0.087091	0.048161	0.01200	0.070000	0.07900	0.090000	0.6110
free.sulfur.dioxide	1000.0	15.862500	10.255048	1.00000	8.000000	14.00000	21.000000	72.0000
total.sulfur.dioxide	1000.0	46.255500	32.748579	6.00000	23.000000	38.00000	60.000000	289.0000
density	1000.0	0.996713	0.001880	0.99007	0.995567	0.99673	0.997833	1.0032
pH	1000.0	3.306920	0.151222	2.74000	3.210000	3.31000	3.400000	4.0100
sulphates	1000.0	0.654180	0.168285	0.33000	0.550000	0.62000	0.720000	2.0000
alcohol	1000.0	10.413533	1.065383	8.40000	9.500000	10.10000	11.000000	14.9000
quality	1000.0	5.625000	0.803131	3.00000	5.000000	6.00000	6.000000	8.0000

```
missing_values_count = data.isnull().sum()
print(missing_values_count)
```

```
fixed.acidity           0
volatile.acidity        0
citric.acid             0
residual.sugar          0
chlorides               0
free.sulfur.dioxide     0
total.sulfur.dioxide    0
density                 0
pH                      0
sulphates               0
alcohol                 0
quality                 0
dtype: int64
```

```
total_cells = np.product(data.shape)
total_missing = data.isnull().sum().sum()
print((total_missing / total_cells) * 100)
```

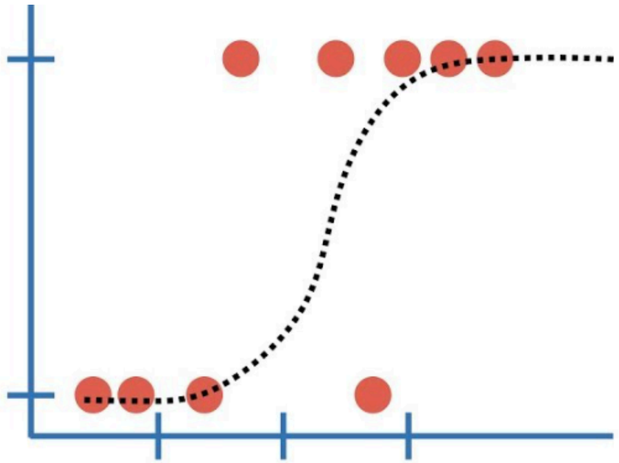
```
0.0
```

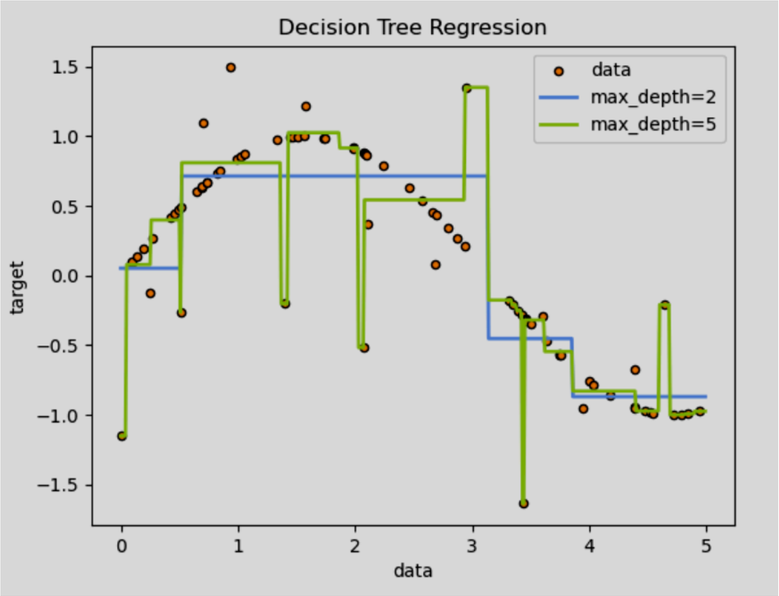
```
total_cells = np.product(data.shape)
total_missing = data.isnull().sum().sum()
print((total_missing / total_cells) * 100)
```

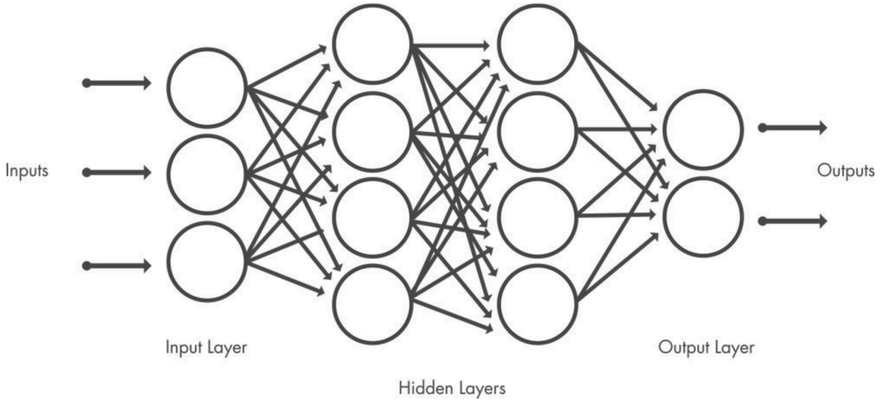
```
0.0
```

Missing Attribute Values: None

Models Tested





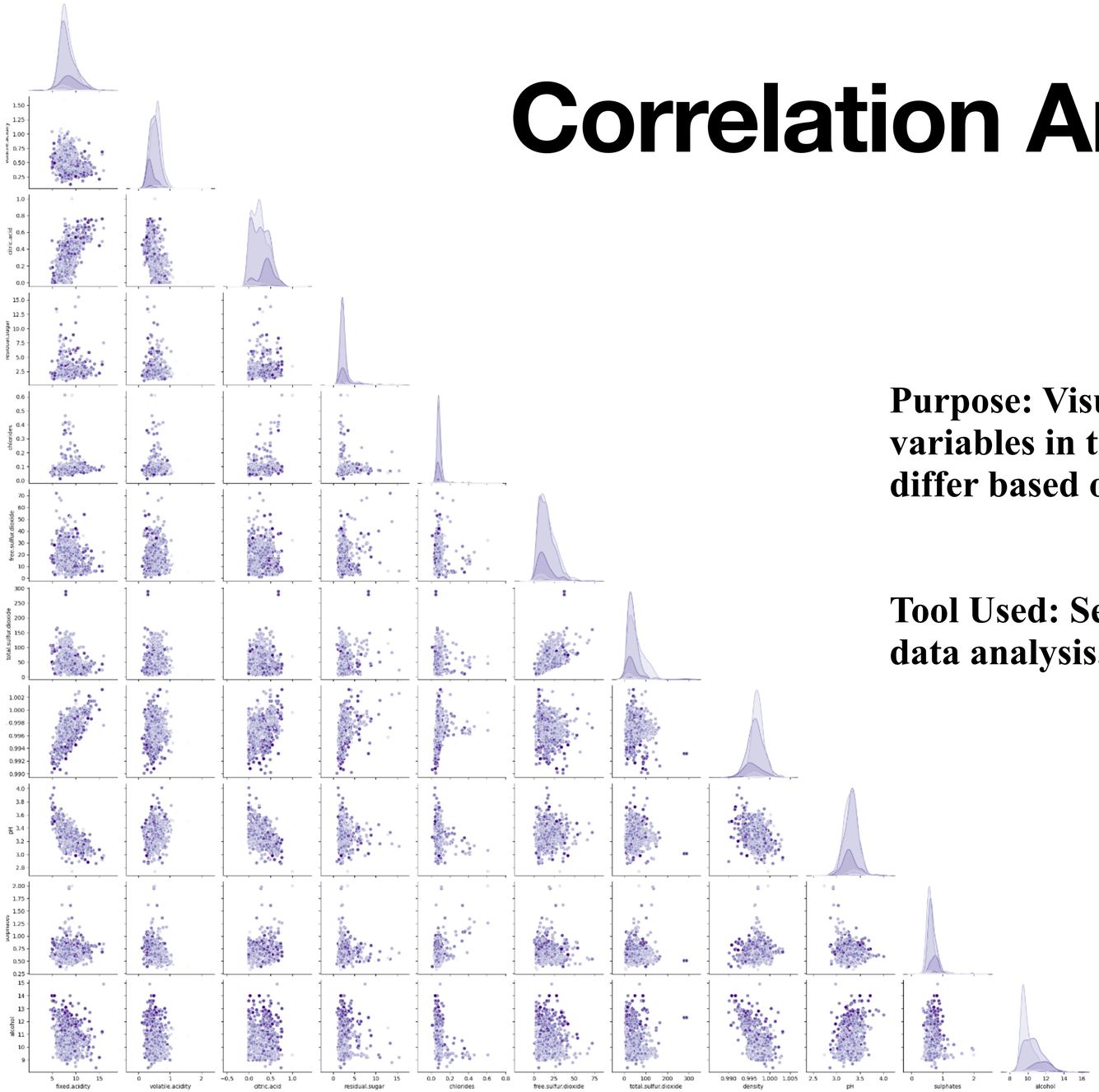


Logistic Regression

Decision Tree

CNN

Correlation Analysis



Purpose: Visualize the pairwise relationships and distributions of variables in the dataset, with a specific focus on how these relationships differ based on the quality attribute.

Tool Used: Seaborn's pairplot function, a powerful tool for exploratory data analysis.

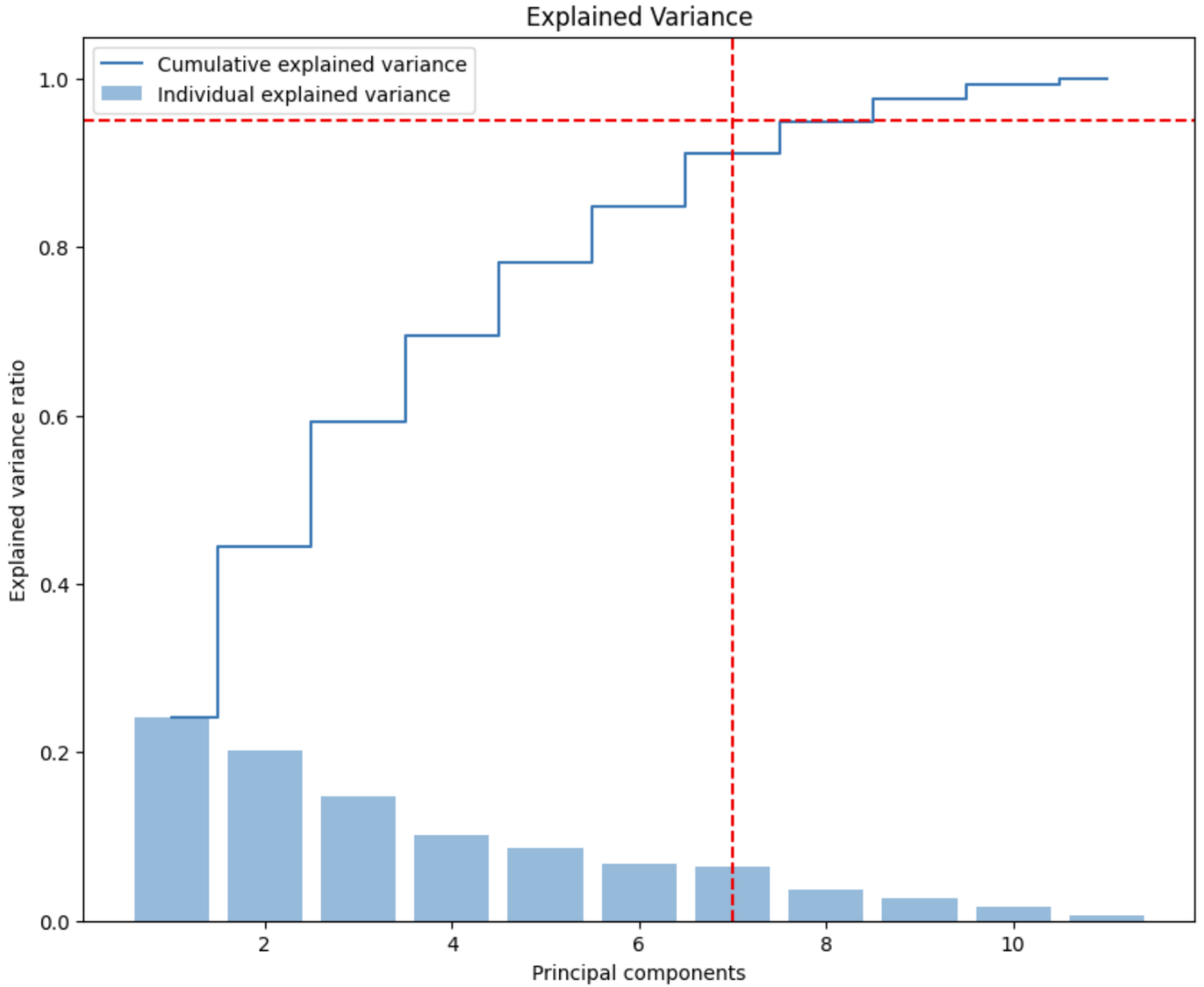
Principle Components Analysis

Is the final result really related to each factor? Or is it that there will be some factors that will dominate the final result

	fixed.acidity	volatile.acidity	citric.acid	residual.sugar	chlorides	free.sulfur.dioxide	total.sulfur.dioxide	density	pH
0	9.7	0.690	0.32	2.5	0.088	22.0	91.0	0.99790	3.29
1	6.6	0.580	0.02	2.4	0.069	19.0	40.0	0.99387	3.38
2	9.2	0.755	0.18	2.2	0.148	10.0	103.0	0.99690	2.87
3	9.0	0.785	0.24	1.7	0.078	10.0	21.0	0.99692	3.29
4	10.6	0.360	0.57	2.3	0.087	6.0	20.0	0.99676	3.14

	Component 1	Component 2	Component 3	Component 4	Component 5	Component 6	Component 7	Component 8	Component 9
0	1.267577	-2.450292	1.118750	0.141006	-0.593554	-0.265828	0.956404	0.563963	1.032002
1	0.079424	-0.060887	-1.374047	-0.415509	0.567828	0.082791	1.633321	0.574262	0.454377
2	3.644562	1.530884	-0.707682	0.335126	-0.062291	0.524247	0.298322	0.102173	0.291330
3	-2.852652	0.986146	-0.551607	-1.948117	-0.874598	-0.664762	-0.617848	-0.378416	-0.649127
4	-3.082605	1.906569	1.514518	-0.061115	-0.265029	-0.608652	0.121202	0.854846	-0.225306

Principle Components Analysis



Mitigate the Impact of Outliers

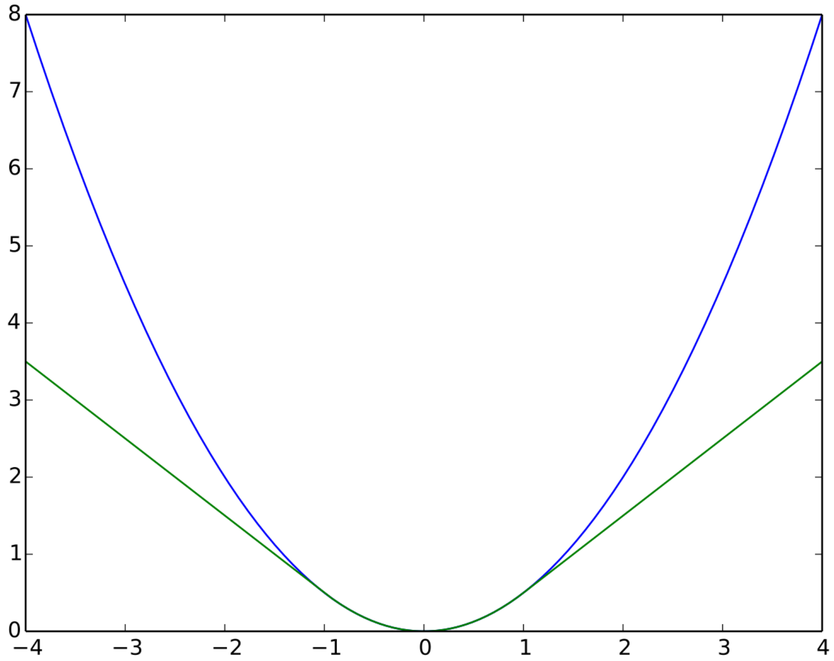
Taking care of most common datas or taking of all the data features? (Like Huber Norm, trying to decrease the impact of outliers)

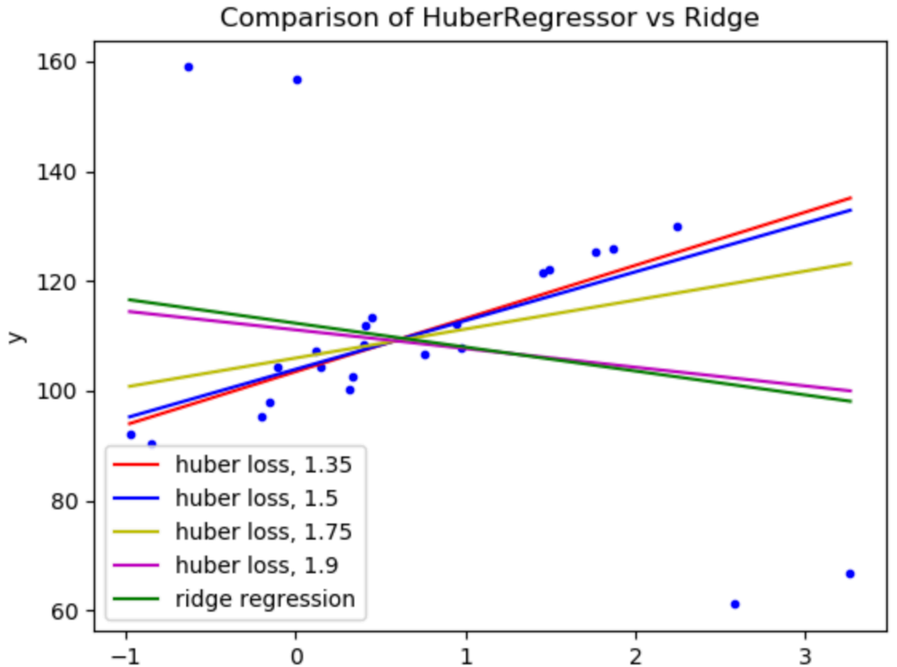
$$\text{Huber}(y, \hat{y}) = \begin{cases} \frac{1}{2}(y_i - \hat{y}_i)^2 & \text{if } |y_i - \hat{y}_i| \leq \delta \\ \delta |y_i - \hat{y}_i| - \frac{1}{2}\delta^2 & \text{otherwise} \end{cases}$$

$$L1(y, \hat{y}) = \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

$$L2(y, \hat{y}) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Huber Norm

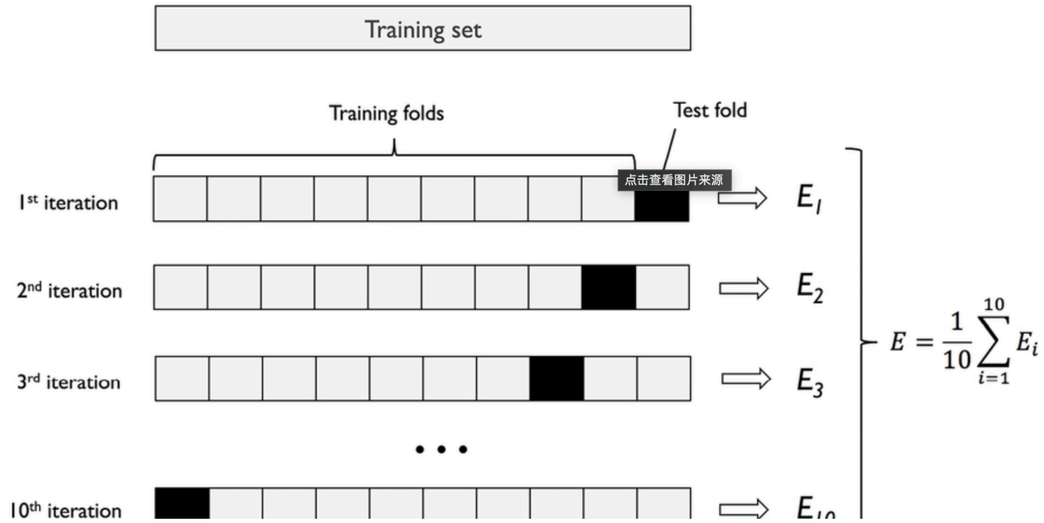




Model Results

We test the model by k-fold cross-validation and submitted to Kaggle, the results are as follows

k=15,
the accuracy is 0.887534



$$\log L(\theta) = n \log\left(\frac{1}{\pi\theta^2}\right) = n\left(-\log(\pi\theta^2)\right) = -n\log(\pi) - 2n\log(\theta)$$