Safe Driver Prediction

XIANZHI LIU, QINGQUAN SONG

Introduction & Motivation

Traffic Safety

- require more tests for those with high potential of causing traffic accident
- regular courses with different frequencies

Insurance expenditures

- People pay almost the same insurance fees nowadays
- charge fees according to the probability

Problem Formulation

Input:

samples $\{X_i, y_i\}$

X_i contains 54 features (with invalid data, both can be categorial and continuous)

y_i can be {0, 1}, 0 indicates the driver i didn't claim insurance last year, 1 indicates otherwise

Output: for each unknow driver with feature X_j predict $P(y_j=1)$

Proposed Solution

Logistic Regression

Neural Networks

SVM, with ‘rbf’ kernel

Xgboost

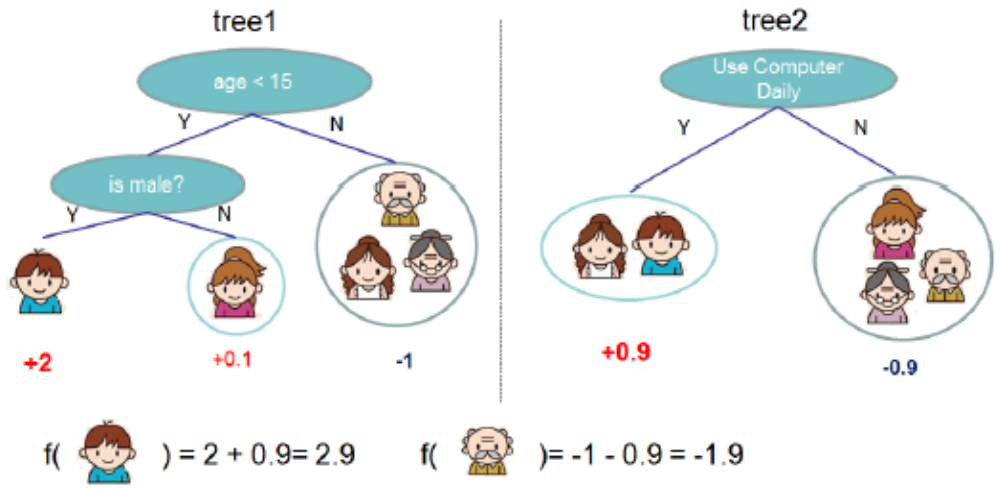
- scalability in all scenarios
- good at dealing with very unbalanced data

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.fc1 = nn.Linear(54, 30)
        self.fc2 = nn.Linear(30, 20)
        self.fc3 = nn.Linear(20, 10)
        self.fc4 = nn.Linear(10, 2)

    def forward(self, x):
        x = x.view(-1, 54)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = F.relu(self.fc3(x))
        x = self.fc4(x)
        return x
```

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.dp=torch.nn.Dropout(p=0.5)
        self.fc1 = nn.Linear(54, 30)
        self.fc2 = nn.Linear(30, 20)
        self.fc3 = nn.Linear(20, 10)
        self.fc4 = nn.Linear(10, 2)
        self.lru = nn.LeakyReLU(0.1)

    def forward(self, x):
        x = x.view(-1, 54)
        x = F.relu(self.dp(self.fc1(x)))
        x = self.lru(self.fc2(x))
        x = F.relu(self.dp(self.fc3(x)))
        x = self.fc4(x)
        return x
```

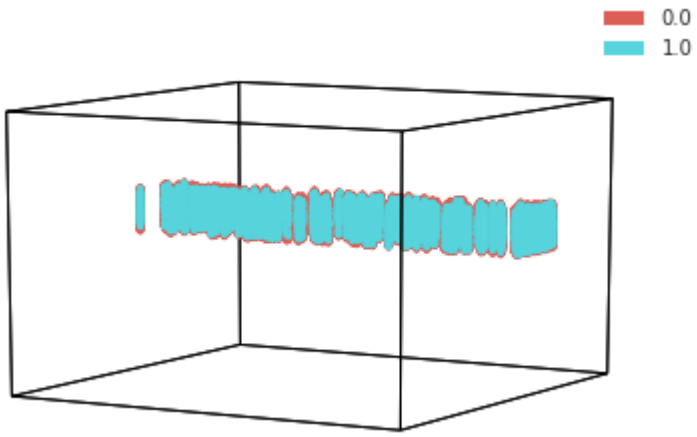


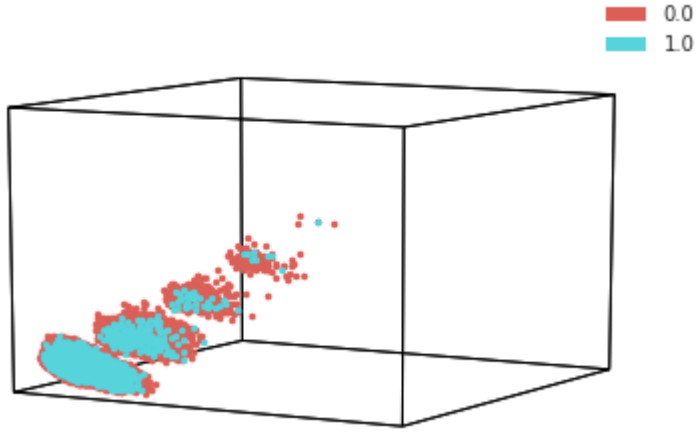
Data description

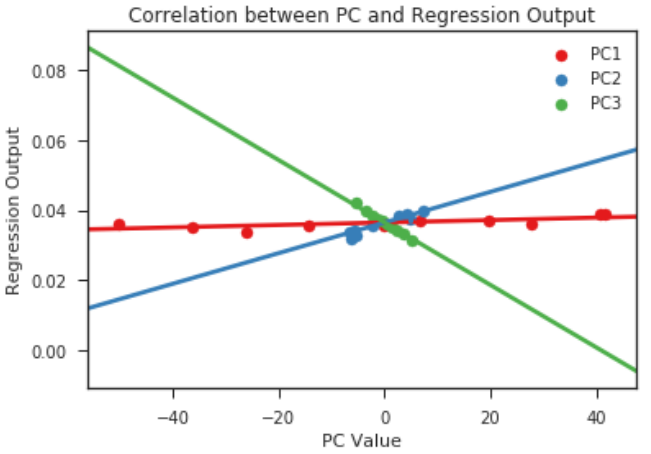
54 features (892816 test samples, 595212 training samples), with invalid data

- remove features with more than 10% invalid data
- generate random value for remaining features with invalid data
- 96.35% negative, 3.65% positive

Data Distribution and PCA



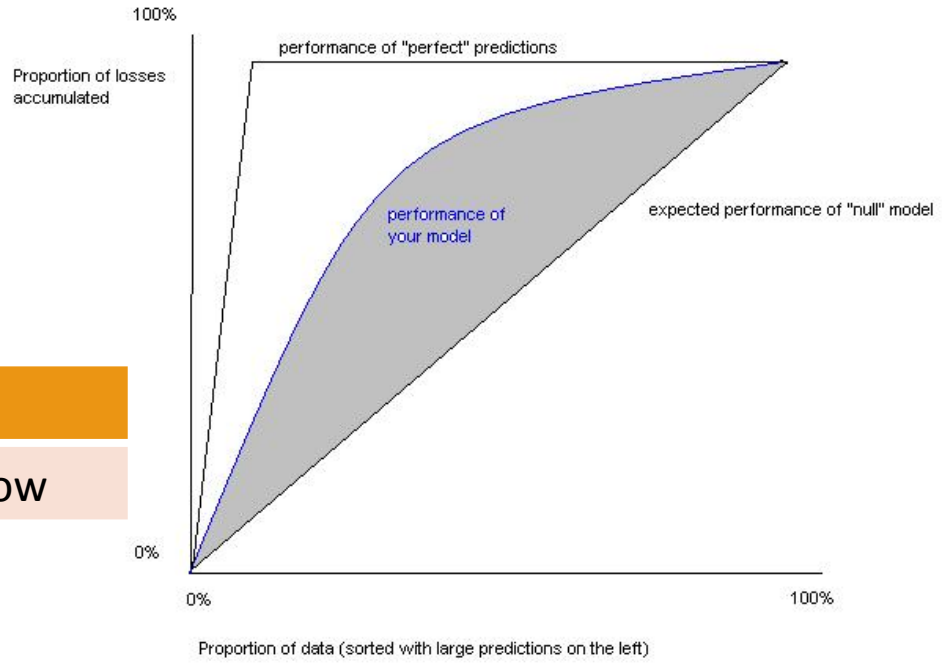




Results & Discussion

Evaluation Metrix: Normalized Gini Coefficient

For leftmost X%, y% loss accumulated



Result:

Logistic	NN1	NN2	10-depXgboost	SVM
0.24124	-0.00340	-0.00974	0.24388	Too slow

Top score of contest: 0.29698