CS 5331 Special Problems in Computer Science: Applied Data Science

Gathering and Processing Data --Model Evaluation and Refinement

Dr. Song Liao
Assistant Professor

September 13, 2024

Model Evaluation

Model Evaluation

- In-sample evaluation tells us how well our model will fit the data used to train it
- Problem?
  - It doesn’t tell us how well the trained model can be used to predict new data
- Solution
  - In-sample data or training data
  - Out-of-sample evaluation or test set

Training/Testing set

- Split dataset into:
  - Training set (70%):
  - Testing set(30%):

- Build and train the model with a training set
- Use testing set to assess the performance of a predictive model

Function train_test_split()

```
y_data = df['price']
```

```
x_data=df.drop('price',axis=1)
```

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=1)
```

- x_data: features or independent variables
- y_data: dataset target: df['price']
- x_train, y_train: parts of available data as training set
- x_test, y_test: parts of available data as testing set
- test_size: percentage of the data for testing (here 30%)

Function train_test_split()

```
lre.fit(x_train[['horsepower']], y_train)
```

```
lre.score(x_test[['horsepower']], y_test)
```

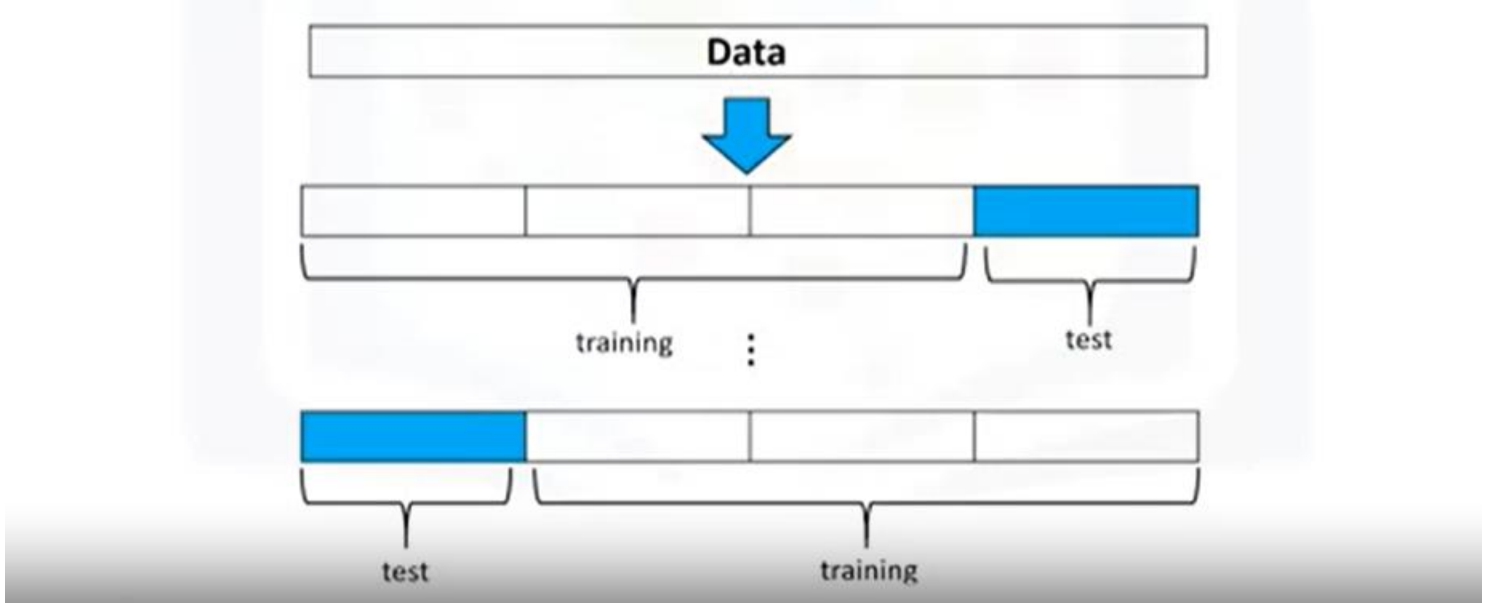
```
0.6287485044222195
```

```
lre.score(x_train[['horsepower']], y_train)
```

```
0.662806312023326 5
```

Cross Validation

- Most common out-of-sample evaluation metrics
- Most effective use of data (each observation is used for both training and testing)



Cross Validation

```python
from sklearn.model_selection import cross_val_score
```

```python
Rcross = cross_val_score(lre, x_data[['horsepower']], y_data, cv=4)
```

```python
Rcross
```

```
array([0.7746232 , 0.51716687, 0.74785353, 0.04839605])
```

```python
print("The mean of the folds are", Rcross.mean(), "and the standard deviation is" , Rcross.std())
```

```
The mean of the folds are 0.522009915042119 and the standard deviation is 0.291183944475603
```

Cross Validation

- Function cross_val_predict() returns the prediction that was obtained for each element when it was in the test set

```
from sklearn.model_selection import cross_val_predict
```

```
yhat = cross_val_predict(lre,x_data[['horsepower']], y_data,cv=4)
yhat[0:5]
```

```
array([14141.63807508, 14141.63807508, 20814.29423473, 12745.03562306,
       14762.35027598])
```

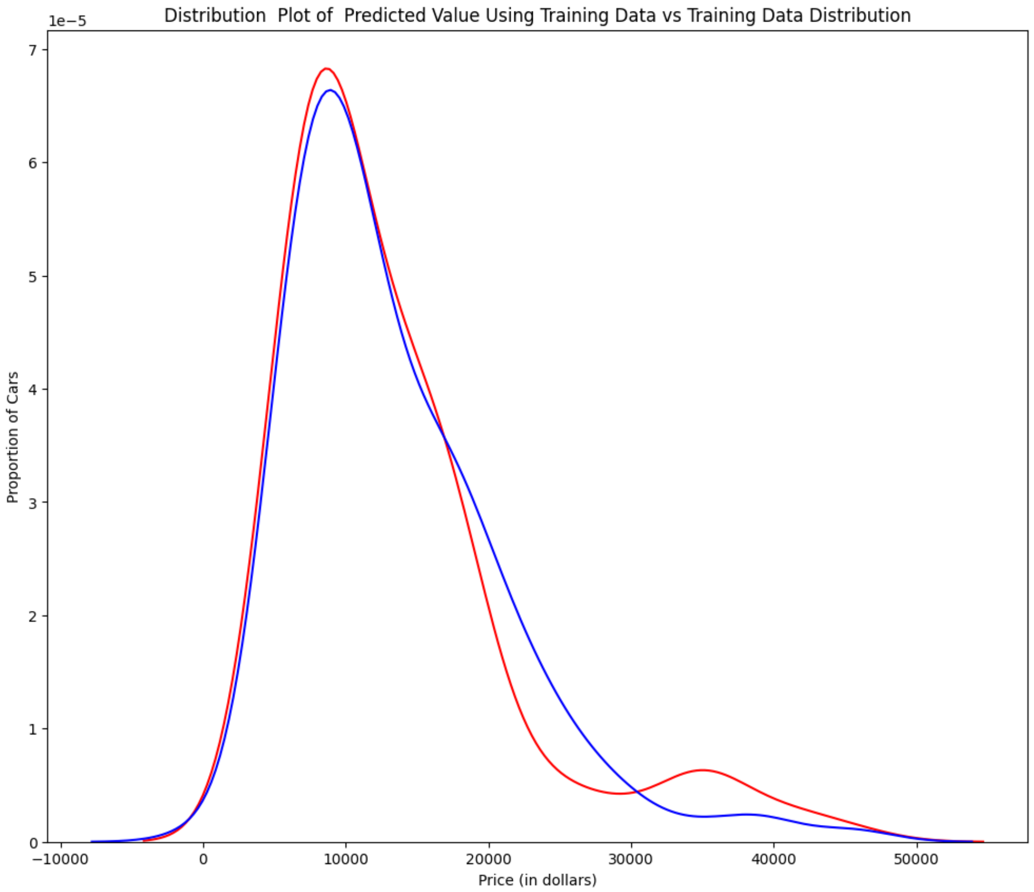
Overfitting

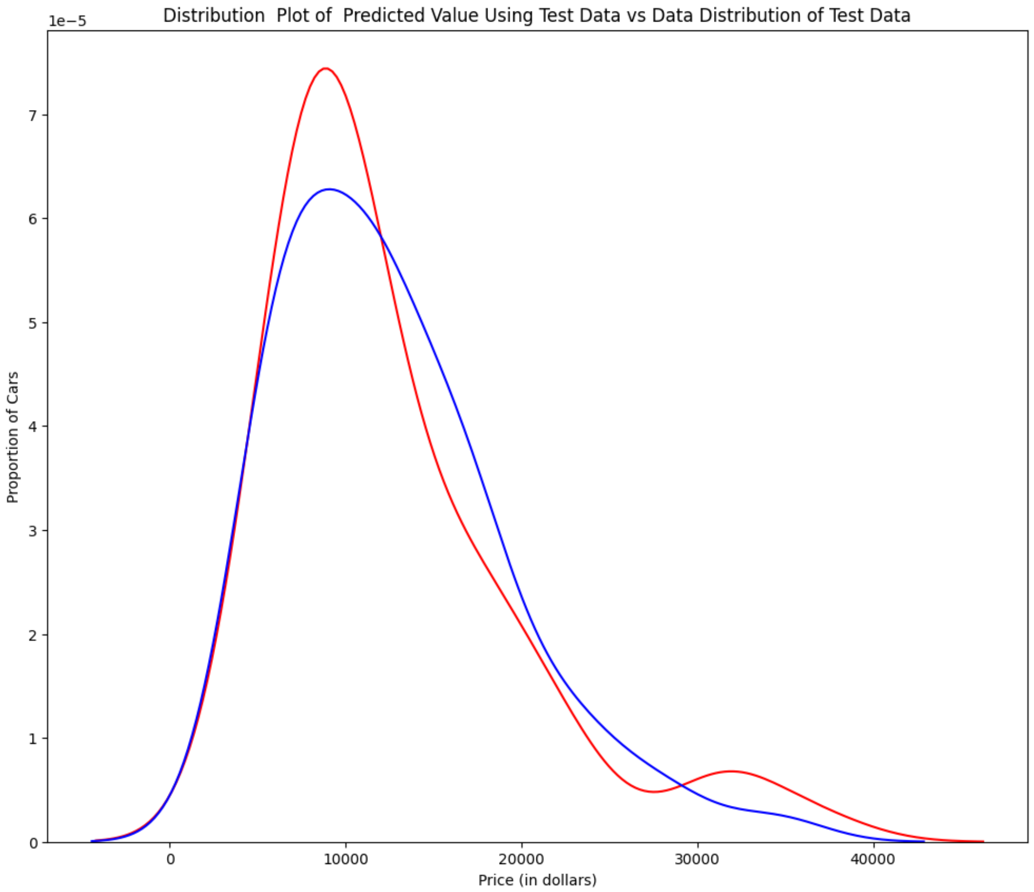
```
lr = LinearRegression()
lr.fit(x_train[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']], y_train)
```

```
yhat_train = lr.predict(x_train[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']])
yhat_train[0:5]
```

```
Title = 'Distribution  Plot of  Predicted Value Using Training Data vs Training Data Distribution'
DistributionPlot(y_train, yhat_train, "Actual Values (Train)", "Predicted Values (Train)", Title)
```

Overfitting





Overfitting

Overfitting

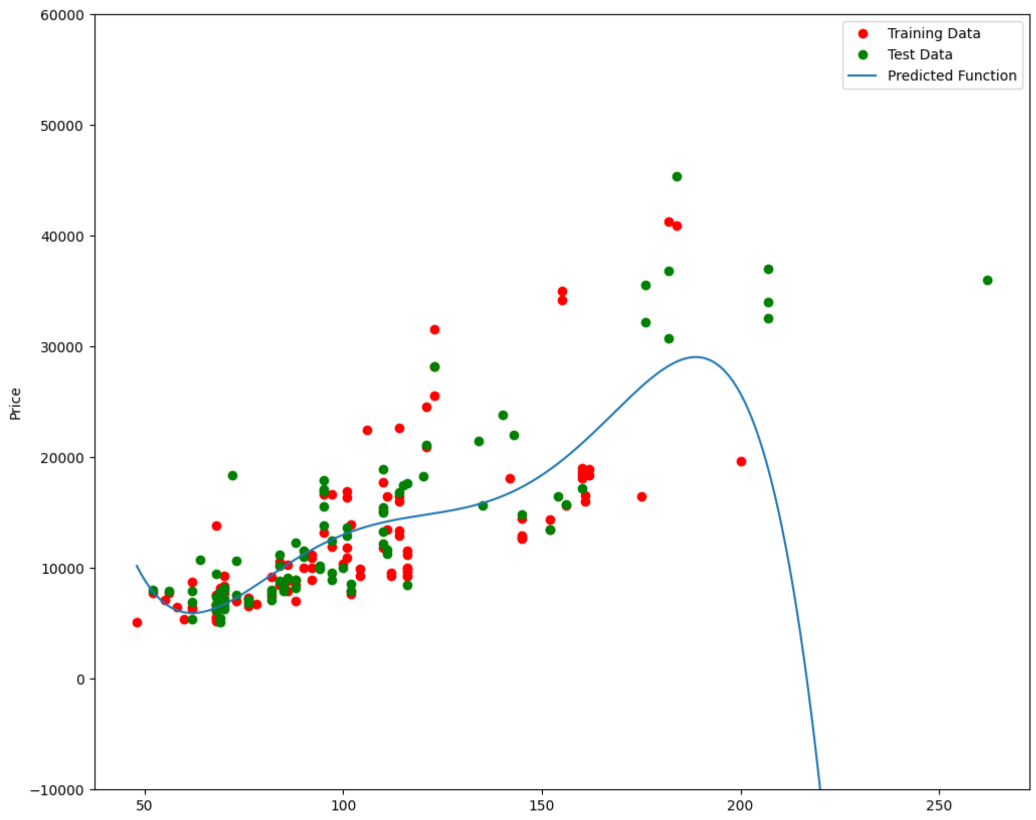
```
pr = PolynomialFeatures(degree=5)
x_train_pr = pr.fit_transform(x_train[['horsepower']])
x_test_pr = pr.fit_transform(x_test[['horsepower']])
pr
```

```
poly.score(x_train_pr, y_train)
```

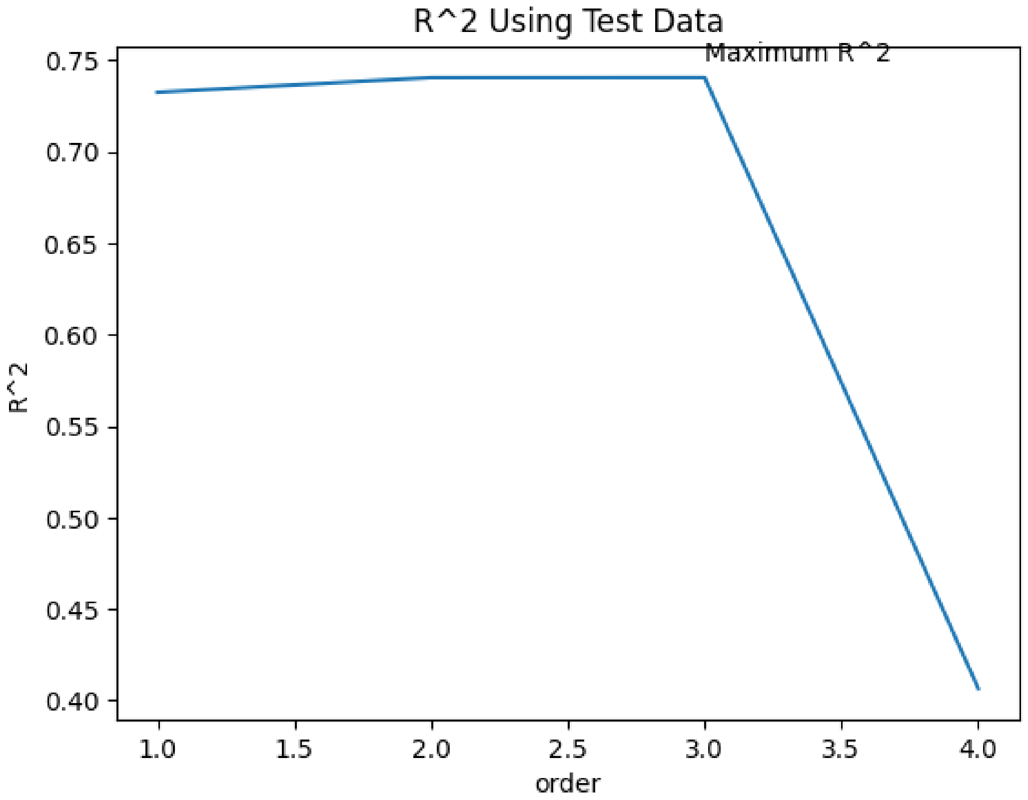
0.5567716902120254

```
poly.score(x_test_pr, y_test)
```

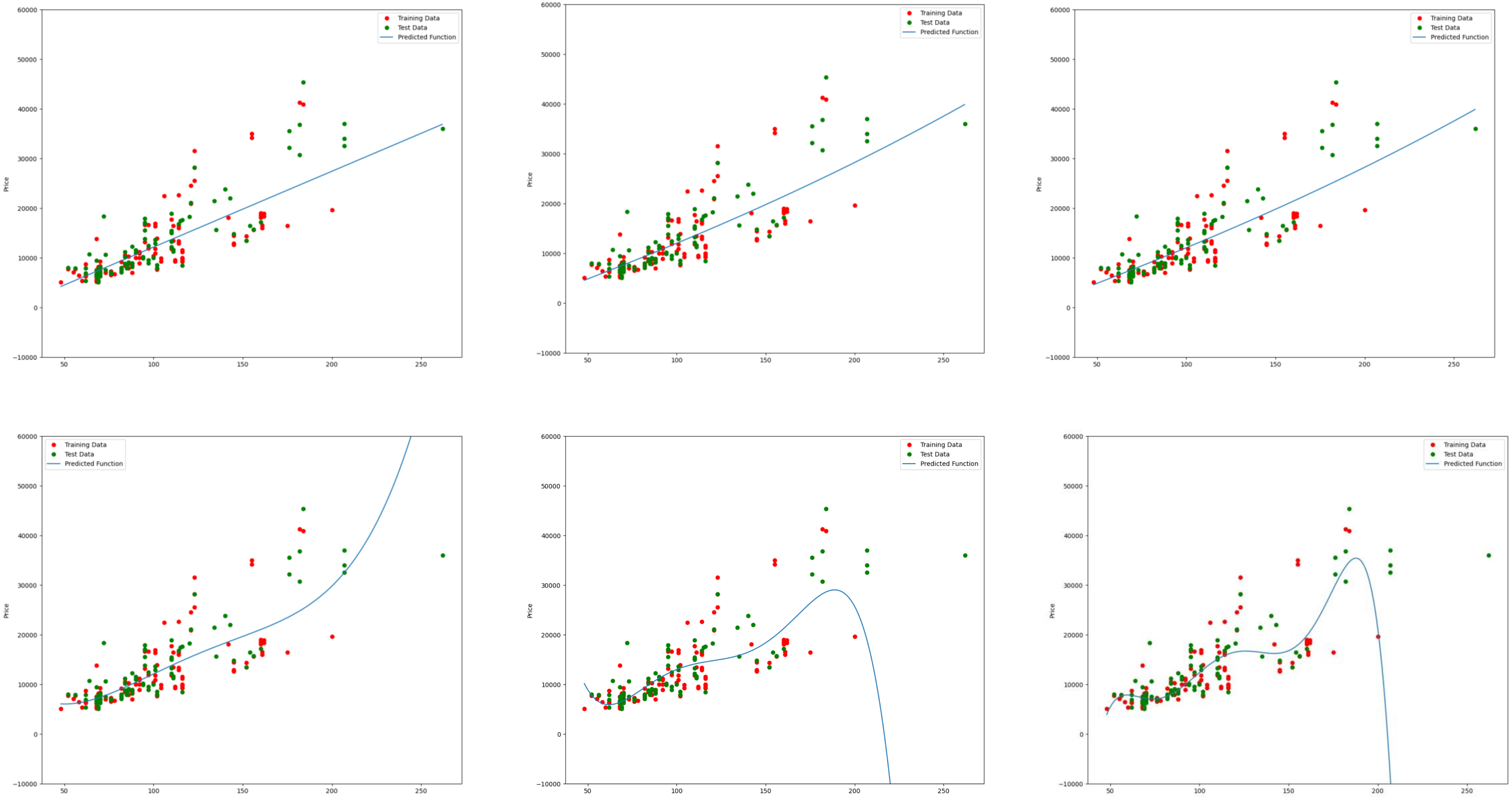
-29.871340302043684



Overfitting



Overfitting



Ridge Regression

- Ridge regression is a type of linear regression that adds a regularization term to prevent overfitting and improve the generalization of the model.

$$\hat{y} = 1 + 2x - 3x^2 - 2x^3 \boxed{- 12x^4 - 40x^5 + 80x^6 + 71x^7 - 141x^8 - 38x^9 + 75x^{10}}$$

Alpha	x	x^2	x^3	x^4	x^5	x^6	x^7	x^8	x^9	x^{10}
0	2	-3	-2	-12	-40	80	71	-141	-38	75
0.001	2	-3	-7	5	4	-6	4	-4	4	6
0.01	1	-2	-5	-0.04	0.15	-1	1	-0.5	0.3	1
1	0.5	-1	-1	-0.614	0.70	-0.38	-0. 56	-0.21	-0.5	-0.1
10	0	-0.5	-0.3	-0.37	-0.30	-0.30	-0.22	-0.22	-0.22	-0.17

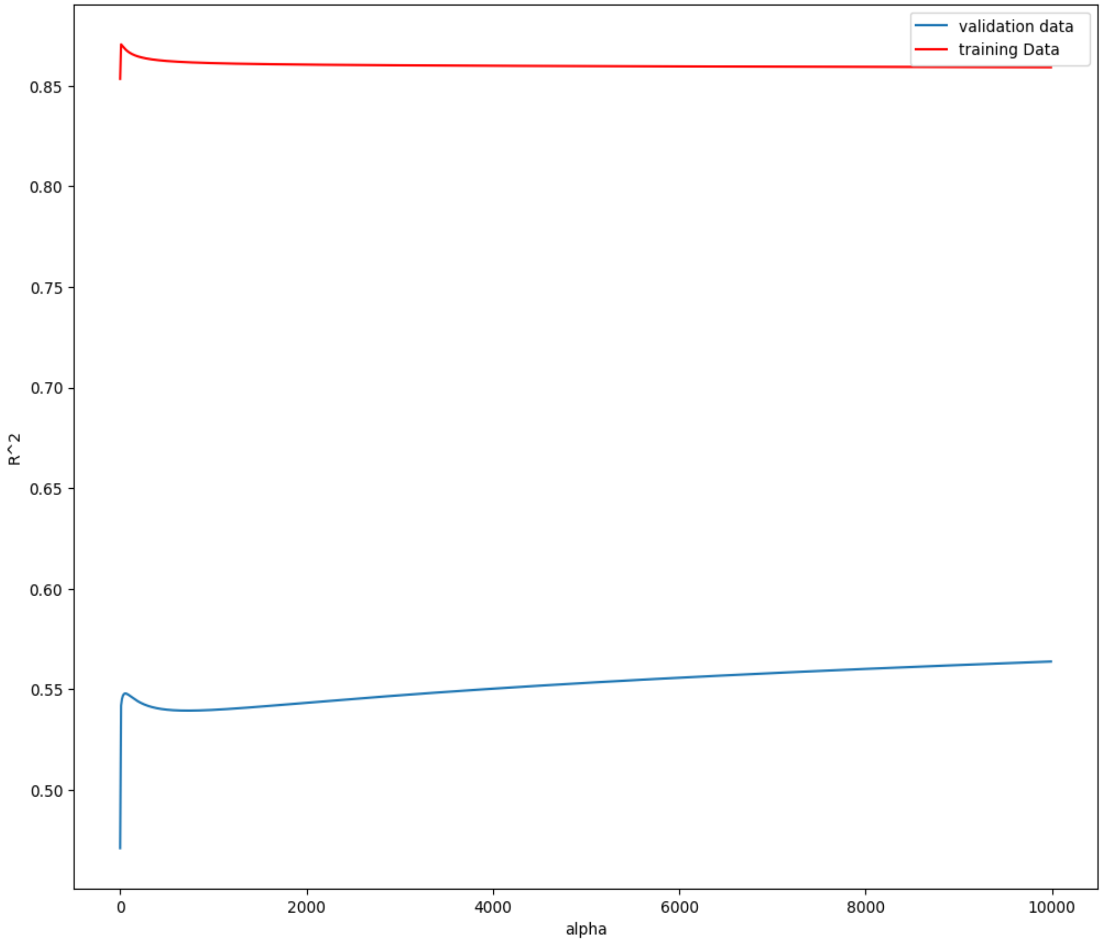
Ridge Regression

```
from sklearn.linear_model import Ridge
```

```
RigeModel=Ridge(alpha=0.1)
```

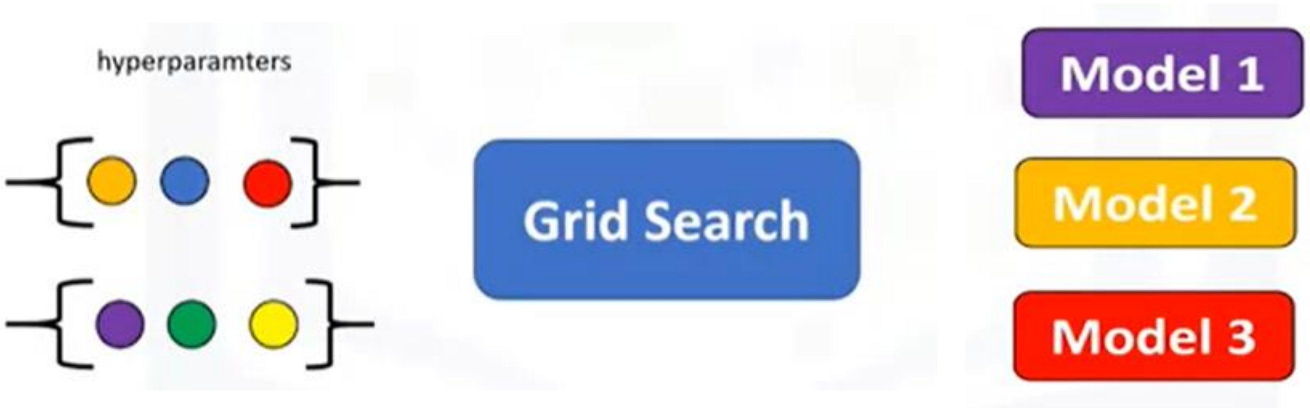
```
RigeModel.fit(x_train_pr, y_train)
```

```
Rsqu_test = []
Rsqu_train = []
dummy1 = []
ALFA = 10 * np.array(range(0,1000))
for alfa in ALFA:
    RigeModel = Ridge(alpha=alfa)
    RigeModel.fit(x_train_pr, y_train)
    Rsqu_test.append(RigeModel.score(x_test_pr, y_test))
    Rsqu_train.append(RigeModel.score(x_train_pr, y_train))
```



Hyperparameters

- The term alpha in Ridge regression is called hyperparameter
- Scikit-learn has a means of automatically iterating over these hyperparameters using cross-validation called Grid Search



Grid Search

```
from sklearn.model_selection import GridSearchCV
```

```
parameters1= [{'alpha': [0.001,0.1,1, 10, 100, 1000, 10000, 100000, 100000]}]
parameters1
```

```
RR=Ridge()
```

```
Grid1 = GridSearchCV(RR, parameters1,cv=4)
```

```
Grid1.fit(x_data[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']], y_data)
```

```
BestRR=Grid1.best_estimator_
BestRR
```

```
▾ Ridge
Ridge(alpha=10000)
```

```
BestRR.score(x_test[['horsepower', 'curb-weight', 'engine-size', 'highway-mpg']], y_test)
```

```
0.8411649831036151
```

Copy Right Notice

- Most slides in this presentation are adopted from various sources. The Copyright belong to the original authors. Thanks!