Hands-on Practice Lab: Model Evaluation and Refinement

Estimated time needed: **45** minutes

In this lab, you will use the skills acquired throughout the module, and try to refine your model's performance in predicting the price of a laptop, given the attribute values.

Objectives

After completing this lab you will be able to:

- Use training, testing and cross validation to improve the performance of the dataset.
- Identify the point of overfitting of a model
- Use Ridge Regression to identify the change in performance of a model based on its hyperparameters
- Use Grid Search to identify the best performing model using different hyperparameters

Setup

For this lab, we will be using the following libraries:

- `skillsnetwork` for downloading the dataset
- `pandas` for managing the data.
- `numpy` for mathematical operations.
- `sklearn` for machine learning and machine-learning-pipeline related functions.
- `seaborn` for visualizing the data.
- `matplotlib` for additional plotting tools.

Installing Required Libraries

The following required libraries are pre-installed in the Skills Network Labs environment. However, if you run this notebook commands in a different Jupyter environment (e.g. Watson Studio or Ananconda), you will need to install these libraries by removing the `#` sign before `%pip` in the code cell below.

The following required libraries are **not** pre-installed in the Skills Network Labs environment. **You will need to run the following cell** to install them:

```
import piplite
await piplite.install('seaborn')
```

Importing Required Libraries

We recommend you import all required libraries in one place (here):

```
from tqdm import tqdm
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from sklearn.model_selection import train_test_split, cross_val_score,
GridSearchCV
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.preprocessing import PolynomialFeatures
```

Importing the Dataset

Run the cell below to donwload the dataset into the console.

```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())

filepath = 'https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/
laptop_pricing_dataset_mod2.csv'

await download(filepath, "laptops.csv")
file_name="laptops.csv"

df = pd.read_csv(file_name, header=0)
```

> Note: This version of the lab is working on JupyterLite, which requires the dataset to be downloaded to the interface.While working on the downloaded version of this notebook on their local machines(Jupyter Anaconda), the learners can simply **skip the steps above,** and simply use the URL directly in the `pandas.read_csv()` function. You can uncomment and run the statements in the cell below.

Import the data set into a data frame.

```
#filepath = 'https://cf-courses-data.s3.us.cloud-object-
storage.appdomain.cloud/IBMDeveloperSkillsNetwork-DA0101EN-Coursera/
laptop_pricing_dataset_mod2.csv'
#df = pd.read_csv(filepath, header=None)
```

Print the value of df.head().

```
df.head()
```

Task 1 : Using Cross validation to improve the model

Divide the dataset into x_data and y_data parameters. Here y_data is the "Price" attribute, and x_data has all other attributes in the data set.

```
# Write your code below and press Shift+Enter to execute
y_data = df['Price']
x_data=df.drop('Price',axis=1)
```

Split the data set into training and testing subests such that you reserve 10% of the data set for testing purposes.

```
# Write your code below and press Shift+Enter to execute
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data,
test_size=0.10, random_state=1)
print("number of test samples :", x_test.shape[0])
print("number of training samples:",x_train.shape[0])
```

Create a single variable linear regression model using "CPU_frequency" parameter. Print the R^2 value of this model for the training and testing subsets.

```
# Write your code below and press Shift+Enter tolre=LinearRegression()
lre=LinearRegression()
lre.fit(x_train[['CPU_frequency']], y_train)
print(lre.score(x_test[['CPU_frequency']], y_test))
print(lre.score(x_train[['CPU_frequency']], y_train))
```

Run a 4-fold cross validation on the model and print the mean value of R^2 score along with its standard deviation.

```
# Write your code below and press Shift+Enter to execute
Rcross = cross_val_score(lre, x_data[['CPU_frequency']], y_data, cv=4)
print("The mean of the folds are", Rcross.mean(), "and the standard
deviation is" , Rcross.std())
```

Task 2: Overfitting

Split the data set into training and testing components again, this time reserving 50% of the data set for testing.

```
# Write your code below and press Shift+Enter to execute
x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, 
test_size=0.50, random_state=0)
```

To identify the point of overfitting the model on the parameter "CPU_frequency", you'll need to create polynomial features using the single attribute. You need to evaluate the R^2 scores of the model created using different degrees of polynomial features, ranging from 1 to 5. Save this set of values of R^2 score as a list.

```
# Write your code below and press Shift+Enter to execute
lre = LinearRegression()
Rsqu_test = []
order = [1, 2, 3, 4, 5]
for n in order:
    pr = PolynomialFeatures(degree=n)
    x_train_pr = pr.fit_transform(x_train[['CPU_frequency']])
    x_test_pr = pr.fit_transform(x_test[['CPU_frequency']])
    lre.fit(x_train_pr, y_train)
    Rsqu_test.append(lre.score(x_test_pr, y_test))
```

Plot the values of R^2 scores against the order. Note the point where the score drops.

```
# Write your code below and press Shift+Enter to execute
plt.plot(order, Rsqu_test)
plt.xlabel('order')
plt.ylabel('R^2')
plt.title('R^2 Using Test Data')
```

Task 3 : Ridge Regression

Now consider that you have multiple features, i.e. 'CPU_frequency', 'RAM_GB', 'Storage_GB_SSD', 'CPU_core','OS','GPU' and 'Category'. Create a polynomial feature model that uses all these parameters with degree=2. Also create the training and testing attribute sets.

```
# Write your code below and press Shift+Enter to execute
pr=PolynomialFeatures(degree=2)
x_train_pr=pr.fit_transform(x_train[['CPU_frequency', 'RAM_GB', 
'Storage_GB_SSD', 'CPU_core','OS','GPU', 'Category']])
x_test_pr=pr.fit_transform(x_test[['CPU_frequency', 'RAM_GB', 
'Storage_GB_SSD', 'CPU_core','OS','GPU', 'Category']])
```

Create a Ridge Regression model and evaluate it using values of the hyperparameter alpha ranging from 0.001 to 1 with increments of 0.001. Create a list of all Ridge Regression R^2 scores for training and testing data.

```python
# Write your code below and press Shift+Enter to execute
Rsqu_test = []
Rsqu_train = []
Alpha = np.arange(0.001,1,0.001)
pbar = tqdm(Alpha)

for alpha in pbar:
    RigeModel = Ridge(alpha=alpha)
    RigeModel.fit(x_train_pr, y_train)
    test_score, train_score = RigeModel.score(x_test_pr, y_test), 
RigeModel.score(x_train_pr, y_train)
    pbar.set_postfix({"Test Score": test_score, "Train Score": 
train_score})
    Rsqu_test.append(test_score)
    Rsqu_train.append(train_score)
```

Plot the R^2 values for training and testing sets with respect to the value of alpha

```python
# Write your code below and press Shift+Enter to execute

plt.figure(figsize=(10, 6))
plt.plot(Alpha, Rsqu_test, label='validation data')
plt.plot(Alpha, Rsqu_train, 'r', label='training Data')
plt.xlabel('alpha')
plt.ylabel('R^2')
plt.ylim(0, 1)
plt.legend()
```

Task 4: Grid Search

Using the raw data and the same set of features as used above, use GridSearchCV to identify the value of alpha for which the model performs best. Assume the set of alpha values to be used as

```
{0.0001, 0.001, 0.01, 0.1, 1, 10}
```

```python
# Write your code below and press Shift+Enter to execute
parameters1= [{'alpha': [0.0001, 0.001, 0.01, 0.1, 1, 10]}]
parameters1
```

Create a Ridge instance and run Grid Search using a 4 fold cross validation.

```
# Write your code below and press Shift+Enter to execute
RR=Ridge()
Grid1 = GridSearchCV(RR, parameters1,cv=4)
```

Fit the Grid Search to the training data.

```
Grid1.fit(x_train[['CPU_frequency', 'RAM_GB', 'Storage_GB_SSD',
'CPU_core', 'OS', 'GPU', 'Category']], y_train)
```

Print the R^2 score for the test data using the estimator that uses the derived optimum value of alpha.

```
# Write your code below and press Shift+Enter to execute
BestRR=Grid1.best_estimator_
print(BestRR.score(x_test[['CPU_frequency', 'RAM_GB',
'Storage_GB_SSD', 'CPU_core','OS','GPU','Category']], y_test))
```

Congratulations! You have completed the lab

Authors

Abhishek Gagneja

Vicky Kuo

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-09-16	0.1	Abhishek Gagneja	Initial Version Created
2023-09-19	0.2	Vicky Kuo	Reviewed and Revised