INTRODUCTION TO MACHINE LEARNING

HOMEWORK II

Feyza Nur SAKA

1521221051

Computer Engineering

Berna KİRAZ

1) **Learning Curve: In this section, I drew a learning curve for the regression problem.**

  a) I visited github.com/ageron/handson-ml2/blob/master/04_training_linear_models.ipynb

  b) I called the plot_learning_curve method with different models using one column of the Mall_Customer dataset.

   i) **Dataset :**

   head()

	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

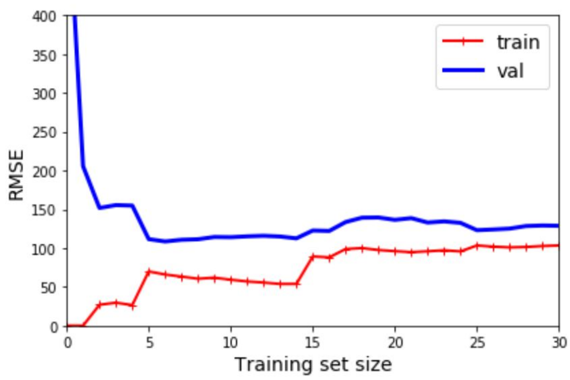
   I used the age column :

   head()

	Age
0	19
1	21
2	20
3	23
4	31

   ii) **Linear Regression :**

```
lin_reg = LinearRegression()
plot_learning_curves(lin_reg, X, y)
plt.axis([0, 30, 0, 400])
save_fig("underfitting_learning_curves_plot")
plt.show()
```

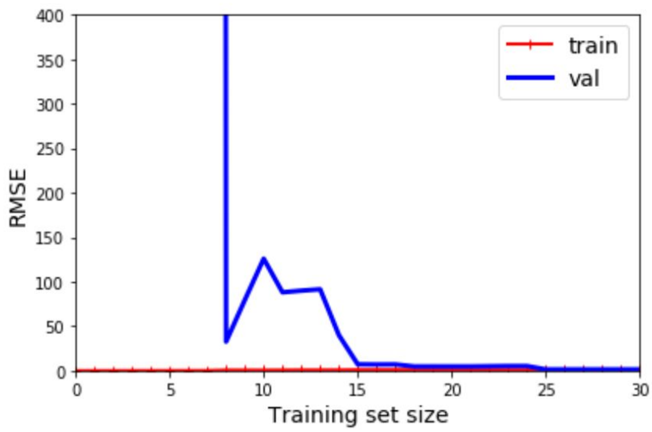


iii) Linear Regression with polynomial features :

```
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures

polynomial_regression = Pipeline([
        ("poly_features", PolynomialFeatures(degree=10, include_bias=False)),
        ("lin_reg", LinearRegression()),
    ])

plot_learning_curves(polynomial_regression, X, y)
plt.axis([0, 30, 0, 400])
save_fig("learning_curves_plot")
plt.savefig("underfitting_learning_curves_plot")
plt.show()
```



c) Comments :

- Overfit status means low bias and high variance.
- High variance pays much attention to training data.
- It cannot predict data that it has not seen before.
- It also gives good results on training data.
- Increasing the polynomial degree would be a better solution.

2) **Parameter Selection: In this part, I applied the Ridge method to the Boston versi set. I used polynomial attributes on the Boston dataset. I found the best values of the alpha parameter in Degree and Ridge method.**

   a) **I uploaded the boston dataset**

```
import pandas as pd
from sklearn.datasets import load_boston
boston = load_boston()
```

   b) **I split the data as train data and test data**

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, random_state = 5)
```

   c) **I scaled the data and computed the polynomial features**

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import Ridge

pipe=make_pipeline(
    StandardScaler(),
    PolynomialFeatures(),
    Ridge())
```

   d) **I tried the values 1, 2, 5, 10 for the grade and 0.001, 0.01, 0.1, 1, 10, 100 for the alpha.**

```
param_grid = {'polynomialfeatures__degree' : [1,2,3],
              'ridge__alpha': [0.001, 0.01, 0.1, 1, 10, 100] }
```

```
from sklearn.model_selection import GridSearchCV
grid = GridSearchCV(pipe,param_grid=param_grid,cv=5,n_jobs=-1)
grid.fit(X_train,y_train)
```

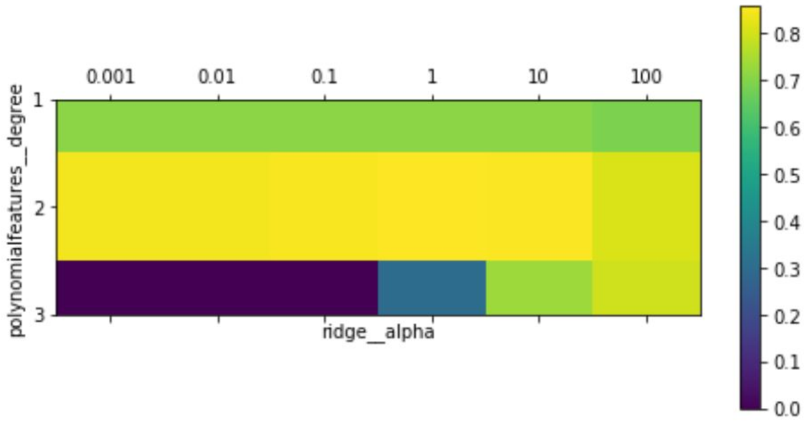
```
Out[9]: GridSearchCV(cv=5, error_score=nan,
             estimator=Pipeline(memory=None,
                                steps=[('standardscaler',
                                        StandardScaler(copy=True,
                                                       with_mean=True,
                                                       with_std=True)),
                                       ('polynomialfeatures',
                                        PolynomialFeatures(degree=2,
                                                           include_bias=True,
                                                           interaction_only=False,
                                                           order='C')),
                                       ('ridge',
                                        Ridge(alpha=1.0, copy_X=True,
                                              fit_intercept=True, max_iter=None,
                                              normalize=False,
                                              random_state=None, solver='auto',
                                              tol=0.001))],
                                verbose=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'polynomialfeatures__degree': [1, 2, 3],
                         'ridge__alpha': [0.001, 0.01, 0.1, 1, 10, 100]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

e) I visualized the results to see better

```
import numpy as np

plt.matshow(grid.cv_results_['mean_test_score'].reshape(3,-1),vmin=0,cmap="viridis")
plt.xlabel("ridge__alpha")
plt.ylabel("polynomialfeatures__degree")
plt.xticks(range(len(param_grid['ridge__alpha'])), param_grid['ridge__alpha'])
plt.yticks(range(len(param_grid['polynomialfeatures__degree'])), param_grid['polynomialfeatures__degree'])
plt.colorbar()
```

```
Out[24]: <matplotlib.colorbar.Colorbar at 0x14870837fc8>
```



i) Using polynomials of degree two helps, but that degree three polynomials are much worse than either degree one or two.

ii) Alpha value seems to be better at 1.0.

KAYNAKÇA

https://www.kaggle.com/akram24/mall-customers

https://github.com/ageron/handson-ml2/blob/master/04_training_linear_models.ipynb

http://178.217.173.109/video_lessons/ENGLISH/MACHINE_LEARNING/ENGLISH/pdf/7.pdf