$$\hat{\beta}_{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

with $\mathbf{I}$ being a $p \times p$ identity matrix.

The ordinary least squares result is

$$\hat{\beta}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

To show that the optimal parameters for Ridge regression are given by:

$$\hat{\beta}_{Ridge} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

we need to derive this from the Ridge regression cost function.

The cost function for Ridge regression is given by:

$$C(\beta) = (\mathbf{y} - \mathbf{X}\beta)^T(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta$$

Expand the Cost Function:

$$C(\beta) = (\mathbf{y}^T - \beta^T\mathbf{X}^T)(\mathbf{y} - \mathbf{X}\beta) + \lambda\beta^T\beta$$

$$C(\beta) = \mathbf{y}^T\mathbf{y} - \mathbf{y}^T\mathbf{X}\beta - \beta^T\mathbf{X}^T\mathbf{y} + \beta^T\mathbf{X}^T\mathbf{X}\beta + \lambda\beta^T\beta$$

then combine the terms:

$$C(\beta) = \mathbf{y}^T\mathbf{y} - 2\mathbf{y}^T\mathbf{X}\beta + \beta^T(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\beta$$

Take the Gradient with Respect to β and set equal to zero for optimization:

$$\frac{\partial C(\beta)}{\partial\beta} = -2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\beta = 0$$

$$-2\mathbf{X}^T\mathbf{y} + 2(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\beta = 0$$

$$(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})\beta = \mathbf{X}^T\mathbf{y}$$

Solve for (β):

$$\beta = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y}$$

Thus, we have shown that the optimal parameters for Ridge regression are:

$\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices of dimensions $n \times n$ and $p \times p$, and $\mathbf{\Sigma}$ is an $n \times p$ matrix which contains the singular values only.

First we show that you can write the OLS solutions in terms of the eigenvectors (the columns) of the orthogonal matrix $\mathbf{U}$ as

$$\tilde{\mathbf{y}}_{OLS} = \mathbf{X}\beta = \sum_{j=0}^{p-1} \mathbf{u_j}\mathbf{u_j}^T\mathbf{y}$$

The OLS solution for β is given by:

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

First we can find $\mathbf{X^TX}$:

$$\mathbf{X}^T\mathbf{X} = (\mathbf{U\Sigma V}^T)^T(\mathbf{U\Sigma V}^T) = \mathbf{V\Sigma}^T\mathbf{U}^T\mathbf{U\Sigma V^T} = \mathbf{V\Sigma}^T\mathbf{\Sigma V^T}$$

Since $\mathbf{U}$ is orthogonal, $\mathbf{U}^T\mathbf{U} = \mathbf{I}$.

Inverse of $\mathbf{X}^T\mathbf{X}$:

$$(\mathbf{X}^T\mathbf{X})^{-1} = (\mathbf{V\Sigma}^T\mathbf{\Sigma V^T})^{-1} = \mathbf{V}^{-T}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{V}^{-1} = \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{V}^T$$

Since $\mathbf{V}$ is orthogonal, $\mathbf{V}^{-1} = \mathbf{V}^T$.

Substitute Back into OLS Solution:

$$\hat{\beta} = \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{V}^T\mathbf{X}^T\mathbf{y}$$

$$\hat{\beta} = \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{V}^T\mathbf{V\Sigma^T U^T y}$$

$$\hat{\beta} = \mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

Compute $\tilde{\mathbf{y}}_{OLS}$:

$$\tilde{\mathbf{y}}_{OLS} = \mathbf{X}\hat{\beta} = \mathbf{U\Sigma V}^T\mathbf{V}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

$$\tilde{\mathbf{y}}_{OLS} = \mathbf{U\Sigma}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{\Sigma}^T\mathbf{U}^T\mathbf{y}$$

And since $(\mathbf{\Sigma^T\Sigma}) = diag(\sigma_0^2, \sigma_1^2, \ldots, \sigma_i^2)$

and $(\mathbf{\Sigma^T\Sigma})^{-1} = diag(1/\sigma_0^2, 1/\sigma_1^2, \ldots, 1/\sigma_i^2)$ then $\mathbf{\Sigma}(\mathbf{\Sigma}^T\mathbf{\Sigma})^{-1}\mathbf{\Sigma}^T = \mathbf{I}$, so:

$$\tilde{\mathbf{y}}_{OLS} = \sum_{j=0}^{p-1} \mathbf{u_j}\mathbf{u_j}^T\mathbf{y}$$

Ridge

For Ridge regression, show that the corresponding equation is

$$\tilde{\mathbf{y}}_{Ridge} = \mathbf{X}\beta_{Ridge} = \mathbf{U\Sigma V}^T(\mathbf{V\Sigma^2V^T} + \lambda\mathbf{I})^{-1}(\mathbf{U\Sigma V^T})^T\mathbf{y} = \sum_{j=1}^{p-1}\mathbf{u_j}\mathbf{u_j^T}\frac{\sigma_j^2}{\sigma_j^2+\lambda}\mathbf{y}$$

First we start with:

$$\mathbf{U\Sigma V}^T(\mathbf{V\Sigma^2V^T} + \lambda\mathbf{I})^{-1}(\mathbf{U\Sigma V^T})^T\mathbf{y}$$

and re-write to:

$$\mathbf{U\Sigma V}^T V(\mathbf{\Sigma^2} + \lambda\mathbf{I})^{-1}V^T(\mathbf{U\Sigma V^T})^T\mathbf{y}$$

Then we can further simplify:

$$\boldsymbol{U\Sigma(\Sigma^2+\lambda I)^{-1}\Sigma^T U^T y}$$

Now we can take a look at $\mathbf{\Sigma}(\mathbf{\Sigma}^2 + \lambda\mathbf{I})^{-1}\Sigma^T$ element by element,

$$\mathbf{\Sigma}(\mathbf{\Sigma}^2 + \lambda\mathbf{I})^{-1}\Sigma^T = (\frac{\sigma_0^2}{\sigma_0^2+\lambda}, \frac{\sigma_1^2}{\sigma_1^2+\lambda}, \frac{\sigma_i^2}{\sigma_i^2+\lambda})$$

Final expression:

$$\tilde{\mathbf{y}}_{Ridge} = \mathbf{U\Sigma V}^T V(\mathbf{\Sigma^2} + \lambda\mathbf{I})^{-1}V^T(\mathbf{U\Sigma V^T})^T\mathbf{y} = \sum_{j=1}^{p-1}\mathbf{u_j}\mathbf{u_j}^T\frac{\sigma_j^2}{\sigma_j^2+\lambda}\mathbf{y}$$

Exercise 2: Adding Ridge Regression

In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
```

```
In [ ]: X = np.ones((n,6))
        X[:,1] = x[:,0]
        X[:,2] = x[:,0]**2
        X[:,3] = x[:,0]**3
        X[:,4] = x[:,0]**4
        X[:,5] = x[:,0]**5


        def scaling(X,y):
            xmean = np.mean(X, axis=0)
            xstd = np.std(X, axis=0)
            xstd[xstd == 0] = 1 #avoid division by zero
            ymean = np.mean(y)
            ystd = np.std(y)
            X_scaled = (X - xmean)/xstd
            y_scaled = (y - ymean)/ystd
            X_scaled[:,0] = 1
            return X_scaled, y_scaled, xmean, xstd, ymean, ystd

        X_scaled, y_scaled, xmean, xstd, ymean, ystd = scaling(X,y)
        print(X_scaled.shape)
```

```
(100, 6)
```

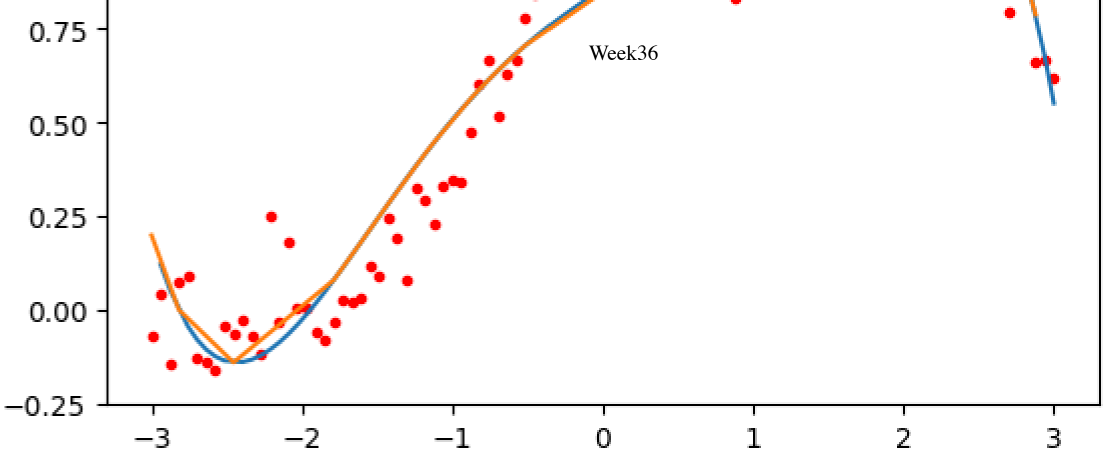
```
In [ ]: #test train split
        X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_scaled, t
        sort = np.argsort(X_train[:,1])
        X_train = X_train[sort]
        y_train = y_train[sort]

        sort = np.argsort(X_test[:,1])
        X_test = X_test[sort]
        y_test = y_test[sort]
```

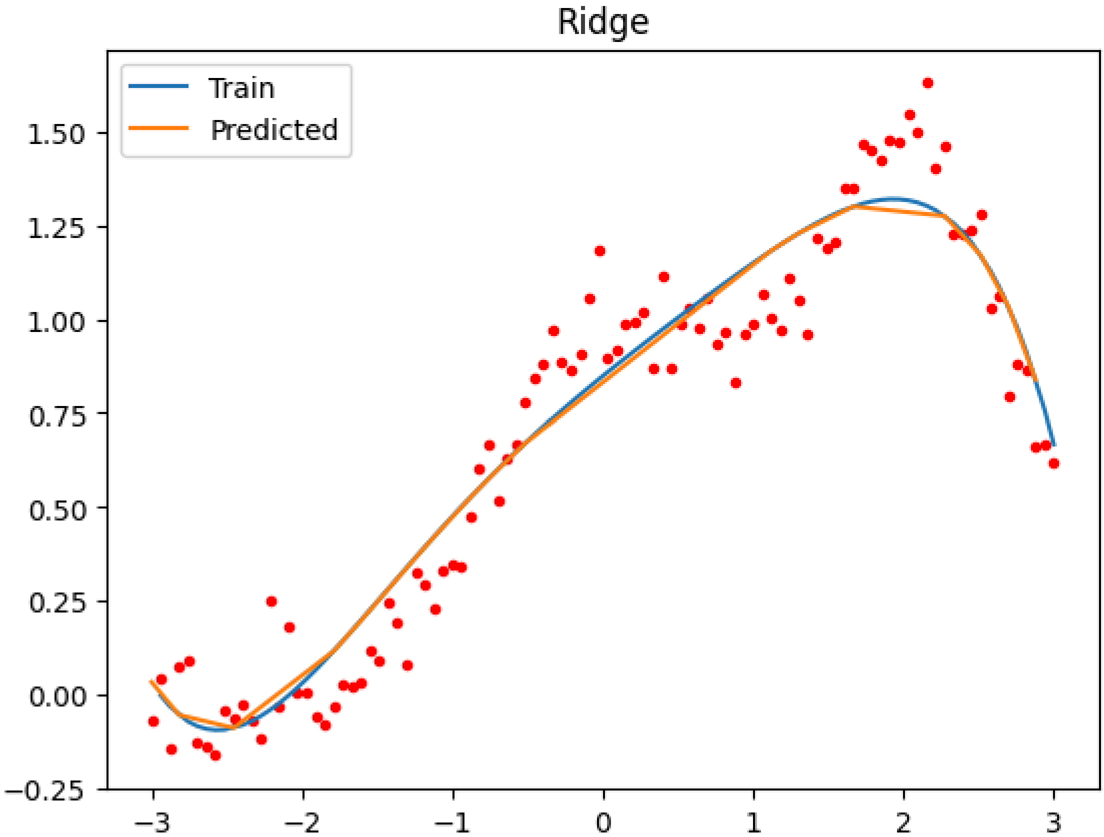
```
In [ ]: #calculate the beta values for OLS and Ridge
        lam = [0.0001, 0.001, 0.01, 0.1, 1]
        betaOLS = []
        betaRidge = []

        for i in lam:
            ols  = np.linalg.inv(X_train.T @ X_train) @ X_train.T @ y_train
            ridge = np.linalg.inv(X_train.T @ X_train + i*np.eye(6)) @ X_train.T

            betaOLS.append(ols)
            betaRidge.append(ridge)
```

```
In [ ]: for i in range(len(lam)):
            ytildeOLS = X_train @ betaOLS[i]
```

```
    print('-------------------------------------')

# #unscale
X_train = X_train*xstd + xmean
X_test = X_test*xstd + xmean
ytildeOLS = ytildeOLS*ystd + ymean
ytildeRidge = ytildeRidge*ystd + ymean
ypredictOLS = ypredictOLS*ystd + ymean
ypredictRidge = ypredictRidge*ystd + ymean


#plot the data
plt.plot(x, y, 'ro',markersize = 3)
plt.plot(X_train[:,1], ytildeOLS, label='Train')
plt.plot(X_test[:,1], ypredictOLS, label='Predicted')
plt.title('OLS')
plt.legend()
plt.show()

#plot the data
plt.plot(x, y, 'ro',markersize = 3)
plt.plot(X_train[:,1], ytildeRidge, label='Train')
plt.plot(X_test[:,1], ypredictRidge, label='Predicted')
plt.title('Ridge')
plt.legend()
plt.show()
```

```
MSE for OLS with   lambda  = None   is |test: 0.0689|  train: 0.0640|
MSE for Ridge with lambda  = 0.0001 is |test: 0.0689|  train: 0.0640|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0689|  train: 0.0640|
MSE for Ridge with lambda  = 0.0010 is |test: 0.0688|  train: 0.0640|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0689|  train: 0.0640|
MSE for Ridge with lambda  = 0.0100 is |test: 0.0684|  train: 0.0640|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0689|  train: 0.0640|
MSE for Ridge with lambda  = 0.1000 is |test: 0.0655|  train: 0.0644|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0689|  train: 0.0640|
MSE for Ridge with lambda  = 1.0000 is |test: 0.0673|  train: 0.0702|
-------------------------------------
```



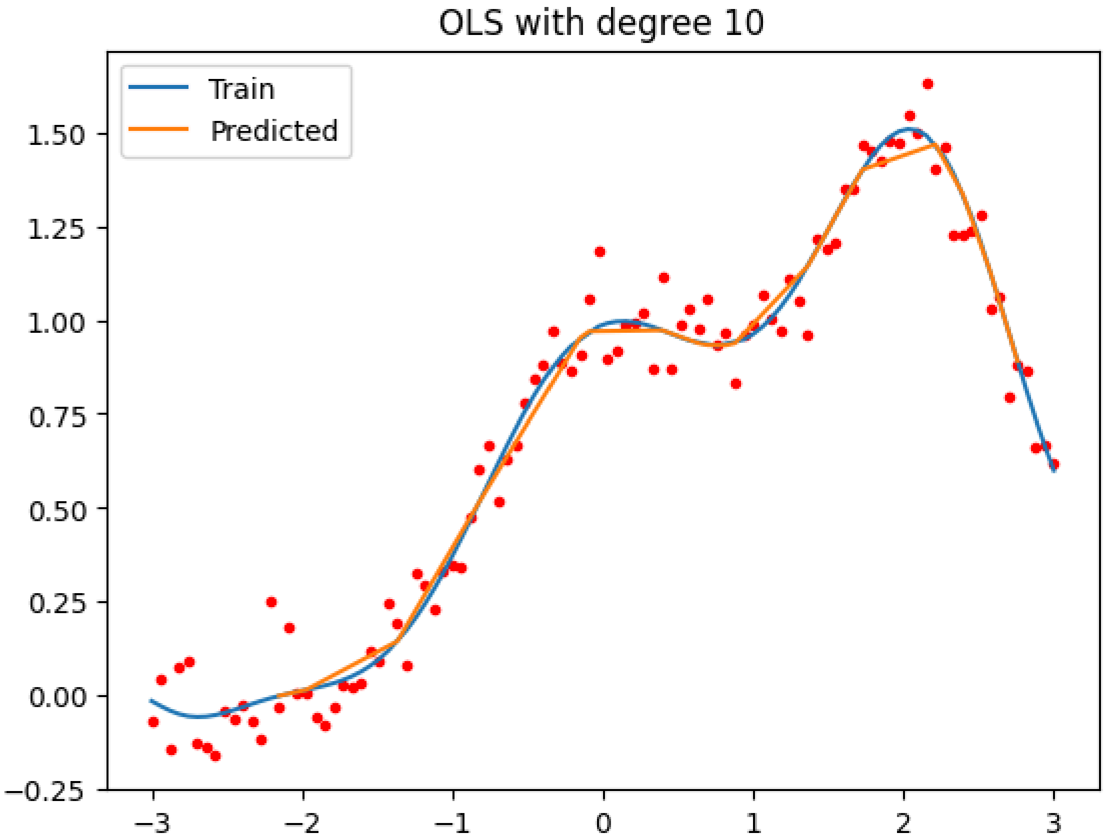


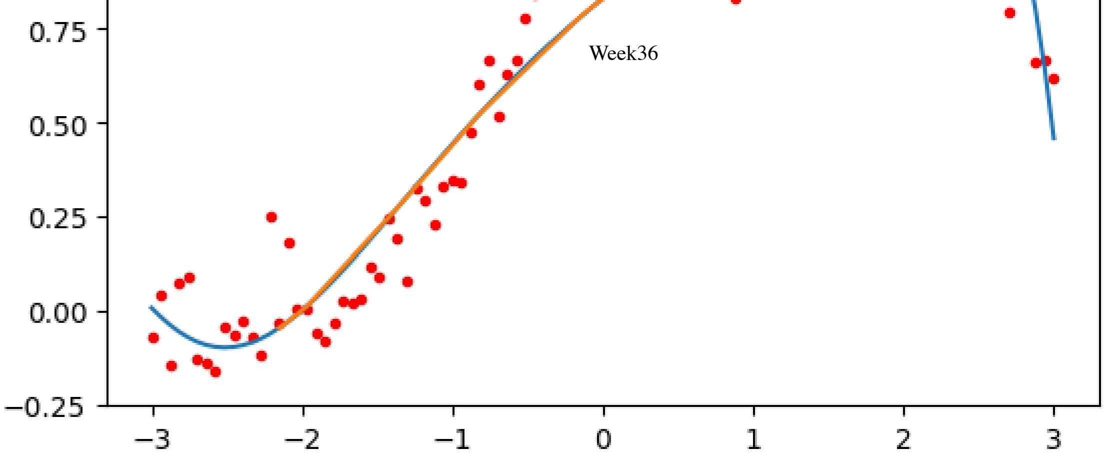
In [ ]:

```
#now we do the same with 10 and 15 degrees
deg = [10, 15]
for d in deg:
    X = PolynomialFeatures(degree=d).fit_transform(x)
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0
```

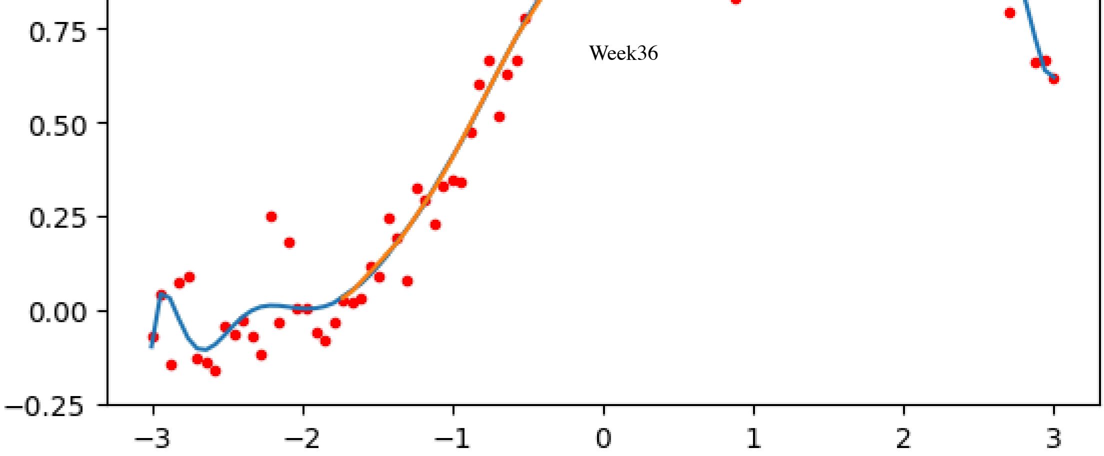
```
    sort = np.argsort(X_test[:,1])
    X_test = X_test[sort]
    y_test = y_test[sort]

    betaOLS = []
    betaRidge = []
    lam = [0.0001, 0.001, 0.01, 0.1, 1]
    for i in lam:
        ols  = np.linalg.inv(X_train.T @ X_train) @ X_train.T @ y_train
        ridge = np.linalg.inv(X_train.T @ X_train + i*np.eye(d+1)) @ X_tr

        betaOLS.append(ols)
        betaRidge.append(ridge)

    for i in range(len(lam)):
        ytildeOLS = X_train @ betaOLS[i]
        ytildeRidge = X_train @ betaRidge[i]

        ypredictOLS = X_test @ betaOLS[i]
        ypredictRidge = X_test @ betaRidge[i]

        MSEOLStest = mse(y_test, ypredictOLS)
        MSERidgetest = mse(y_test, ypredictRidge)
        MSEOLStrain = mse(y_train, ytildeOLS)
        MSERidgetrain = mse(y_train, ytildeRidge)

        print(f'MSE for OLS with   lambda  = None   is |test: {MSEOLStest
        print(f'MSE for Ridge with lambda  = {lam[i]:.4f} is |test: {MSER
        print('------------------------------------')

    # #unscale for plotting
    # for i in range(1, d+1):
    #     X_train[:, i] = X_train[:, i]  + means[i-1]
    #     X_test[:, i] = X_test[:, i]  + means[i-1]

    #unscale using sklearn
    X_train[:,1:] = scaler.inverse_transform(X_train[:,1:])
    X_test[:,1:] = scaler.inverse_transform(X_test[:,1:])


    #plot the data
    plt.plot(x, y, 'ro',markersize = 3)
    plt.plot(X_train[:,1], ytildeOLS, label='Train')
    plt.plot(X_test[:,1], ypredictOLS, label='Predicted')
    plt.title(f'OLS with degree {d}')
    plt.legend()
    plt.show()
```

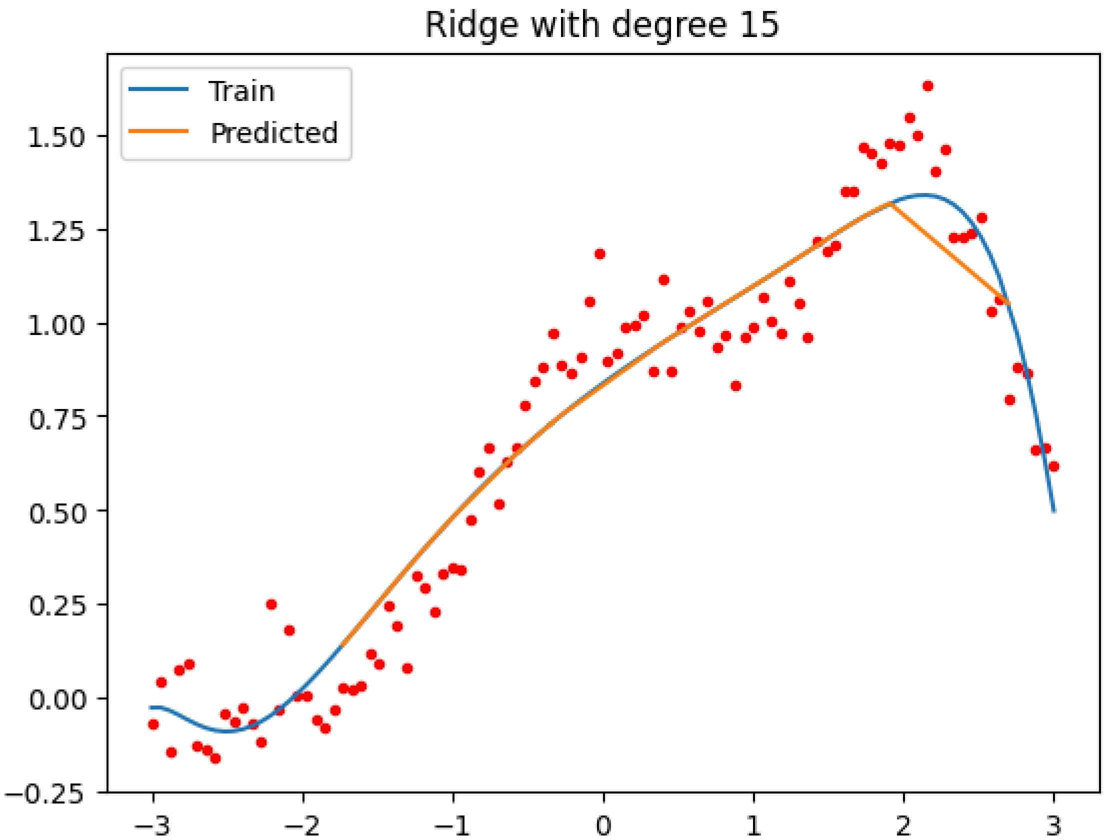
```
MSE for Ridge with lambda  = 0.0010 is |test: 0.0092|  train: 0.0066|
-----------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0086|  train: 0.0060|
MSE for Ridge with lambda  = 0.0100 is |test: 0.0118|  train: 0.0076|
-----------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0086|  train: 0.0060|
MSE for Ridge with lambda  = 0.1000 is |test: 0.0210|  train: 0.0119|
-----------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0086|  train: 0.0060|
MSE for Ridge with lambda  = 1.0000 is |test: 0.0239|  train: 0.0154|
-----------------------------------
```






```
MSE for OLS with   lambda  = None   is |test: 0.0055|  train: 0.0063|
MSE for Ridge with lambda  = 0.0001 is |test: 0.0047|  train: 0.0066|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0055|  train: 0.0063|
MSE for Ridge with lambda  = 0.0010 is |test: 0.0055|  train: 0.0069|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0055|  train: 0.0063|
MSE for Ridge with lambda  = 0.0100 is |test: 0.0088|  train: 0.0084|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0055|  train: 0.0063|
MSE for Ridge with lambda  = 0.1000 is |test: 0.0151|  train: 0.0109|
-------------------------------------
MSE for OLS with   lambda  = None   is |test: 0.0055|  train: 0.0063|
MSE for Ridge with lambda  = 1.0000 is |test: 0.0238|  train: 0.0151|
-------------------------------------
```


0.75
0.50
0.25
0.00
−0.25
−3
−2
−1
0
1
2
3


Ridge with degree 15
Train
Predicted
1.50
1.25
1.00
0.75
0.50
0.25
0.00
−0.25
−3
−2
−1
0
1
2
3