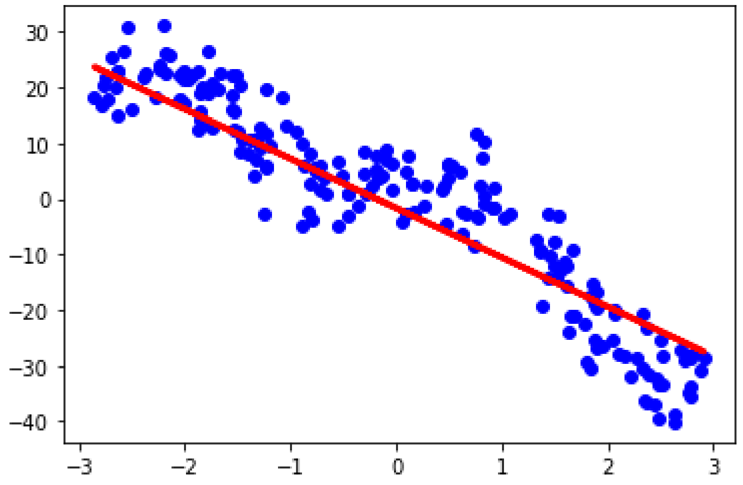
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import mean_squared_error
from sklearn import linear_model
import matplotlib
import matplotlib.pyplot as plt
```

```
m = 500

np.random.seed(seed=5)

X = 6 * np.random.random(m).reshape(-1, 1) - 3

Y = 0.5 * X**5 -5*X**3- X**2 + 2 + 5*np.random.randn(m, 1)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.40, random_state=42)
```

```
# Create linear regression object
model1 = linear_model.LinearRegression()

# Train the model using the training sets
model1.fit(X_train, y_train)

# Make predictions using the testing set
y_pred = model1.predict(X_test)

print('Mean squared error: %.2f'
      % mean_squared_error(y_test, y_pred))

plt.scatter(X_test, y_test, color='blue')

plt.plot(X_test, y_pred, color='red', linewidth=3)
```

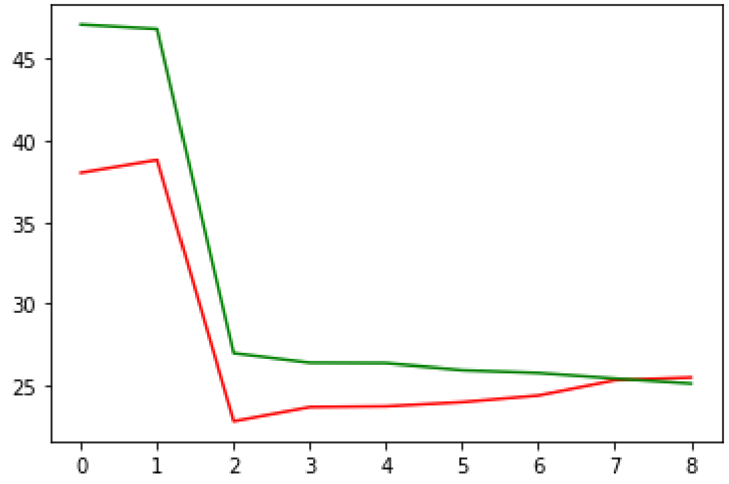
```
Mean squared error: 47.26
[<matplotlib.lines.Line2D at 0x7f9852b69510>]
```



```
from sklearn.preprocessing import PolynomialFeatures
# This code is similar to the code you showed on lecture day 2/22/2022

loss_allModels = []
loss_allModels_train = []
for deg in [2,4,6,8,10,12,14,16,20]:
  poly_features = PolynomialFeatures(degree=deg) #include_bias=True
  X_poly_train = poly_features.fit_transform(X_train)
  X_poly_test = poly_features.fit_transform(X_test)

  # Create linear regression object
  model2 = linear_model.LinearRegression()

  # Train the model using the training sets
  model2.fit(X_poly_train, y_train)

  y_pred2 = model2.predict(X_poly_test)
  y_pred2_train = model2.predict(X_poly_train)

  loss = mean_squared_error(y_test, y_pred2)
  loss_train = mean_squared_error(y_train, y_pred2_train)
```

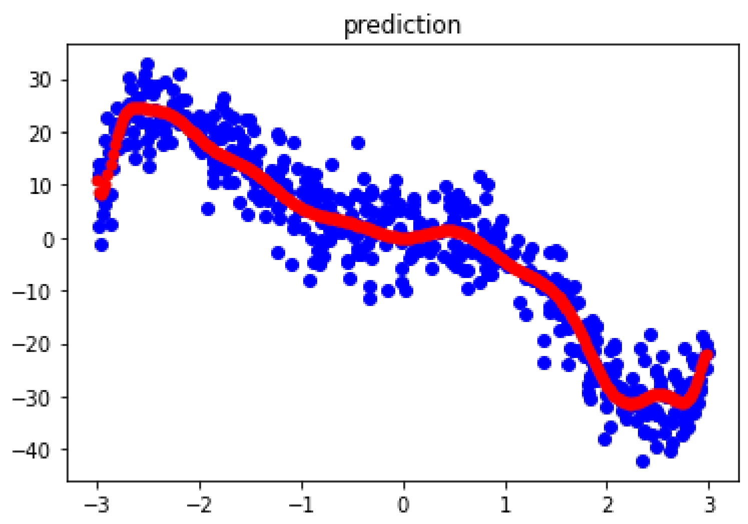
```
    loss_allModels.append(loss)
    loss_allModels_train.append(loss_train)

  plt.figure()
  plt.plot(range(len(loss_allModels)),loss_allModels,color = 'red')
  plt.plot(range(len(loss_allModels_train)),loss_allModels_train,color = 'green')
```

```
[<matplotlib.lines.Line2D at 0x7f9852adbad0>]
```



9)The best degree is 2 because it produces the lowest test loss.

```
Xplot=np.arange(-3.0, 3.0, 0.02)
Xplot=Xplot.reshape(-1, 1)
Xplot_poly = poly_features.fit_transform(Xplot)
yplot_pred = model2.predict(Xplot_poly)

plt.figure()
plt.scatter (X,Y,  color='blue')
plt.scatter (Xplot,yplot_pred, color='red',linewidth=0)
plt.title('prediction')
plt.show()
```



✓ 0s completed at 2:06 PM

✓ 0s completed at 2:00 PM