Advanced Machine Learning Techniques

Gradient Descent, Errors ¶

Accessing the dataset

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [2]:
```
# Access the olympics dataset

oly=pd.read_csv('olympics.csv')
```

In [3]:
```
oly
```

Out[3]:

	Year	Time
0	1896	12.00
1	1900	11.00
2	1904	11.00
3	1908	10.80
4	1912	10.80
5	1920	10.80
6	1924	10.60
7	1928	10.80
8	1932	10.38
9	1936	10.30
10	1948	10.30
11	1952	10.40
12	1956	10.50
13	1960	10.20
14	1964	10.00
15	1968	9.95
16	1972	10.14
17	1976	10.06
18	1980	10.25
19	1984	9.99
20	1988	9.92
21	1992	9.96
22	1996	9.84
23	2000	9.87
24	2004	9.85
25	2008	9.69
26	2012	9.63
27	2016	9.81
28	2021	9.80

Preprocessing

In [4]:

```
X=oly['Year']
X
```

Out[4]:

```
0     1896
1     1900
2     1904
3     1908
4     1912
5     1920
6     1924
7     1928
8     1932
9     1936
10    1948
11    1952
12    1956
13    1960
14    1964
15    1968
16    1972
17    1976
18    1980
19    1984
20    1988
21    1992
22    1996
23    2000
24    2004
25    2008
26    2012
27    2016
28    2021
Name: Year, dtype: int64
```

In [5]:

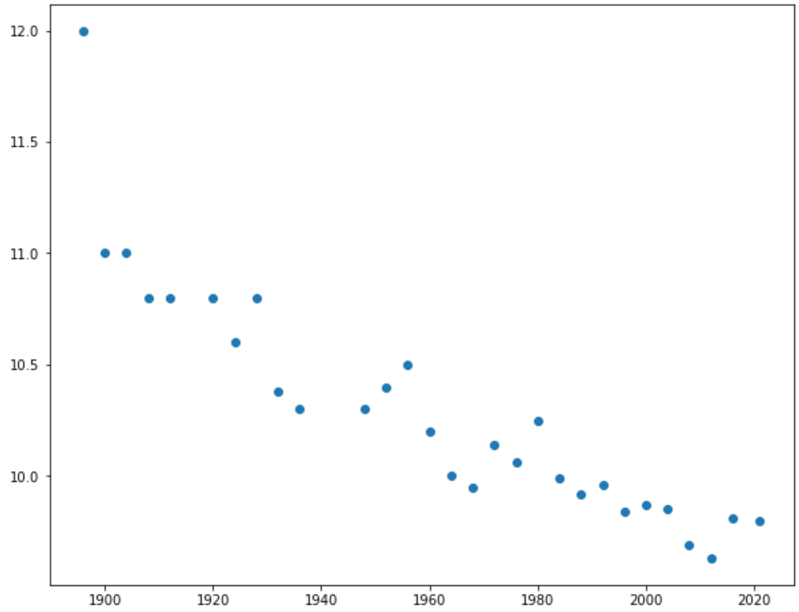
```
y=oly['Time']
y
```

Out[5]:

```
0     12.00
1     11.00
2     11.00
3     10.80
4     10.80
5     10.80
6     10.60
7     10.80
8     10.38
9     10.30
10    10.30
11    10.40
12    10.50
13    10.20
14    10.00
15     9.95
16    10.14
17    10.06
18    10.25
19     9.99
20     9.92
21     9.96
22     9.84
23     9.87
24     9.85
25     9.69
26     9.63
27     9.81
28     9.80
Name: Time, dtype: float64
```

Plotting

In [7]:
```
plt.figure(figsize=(10,8))
plt.scatter(X,y);
```

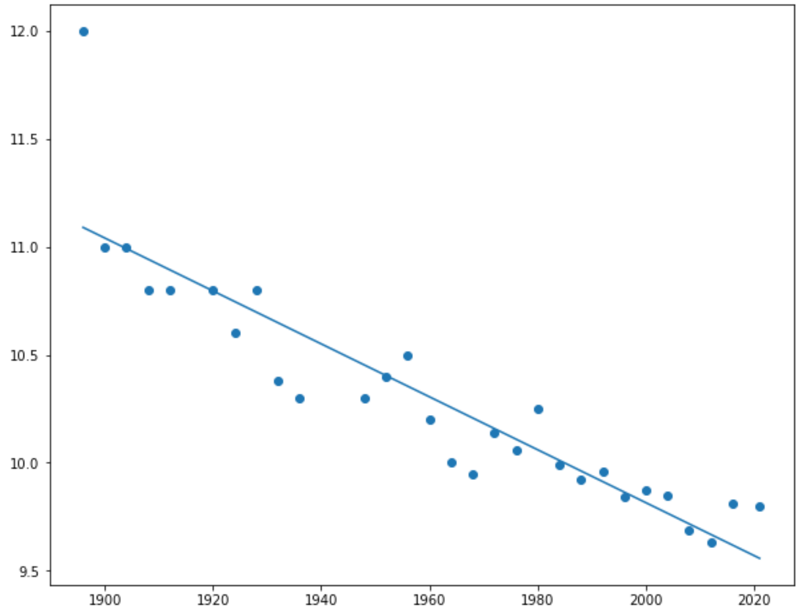


Desinging a simple LR model

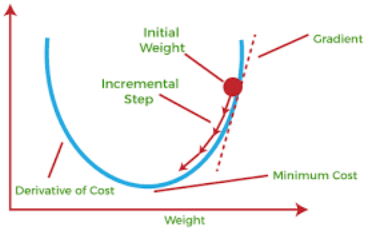
In [10]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()

lr.fit(X.to_numpy().reshape(-1,1),y)
```

Out[10]:
```
LinearRegression()
```

In [11]:
```
# Plotting

plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)));
```



What is behind?



Error:

Bias:

Variance

The model underfits

Polynomial Regression

Degree 2

```
In [12]: from sklearn.preprocessing import PolynomialFeatures

         p2=PolynomialFeatures(degree=2)

         Xp2=p2.fit_transform(X.to_numpy().reshape(-1,1))

         Xp2
```

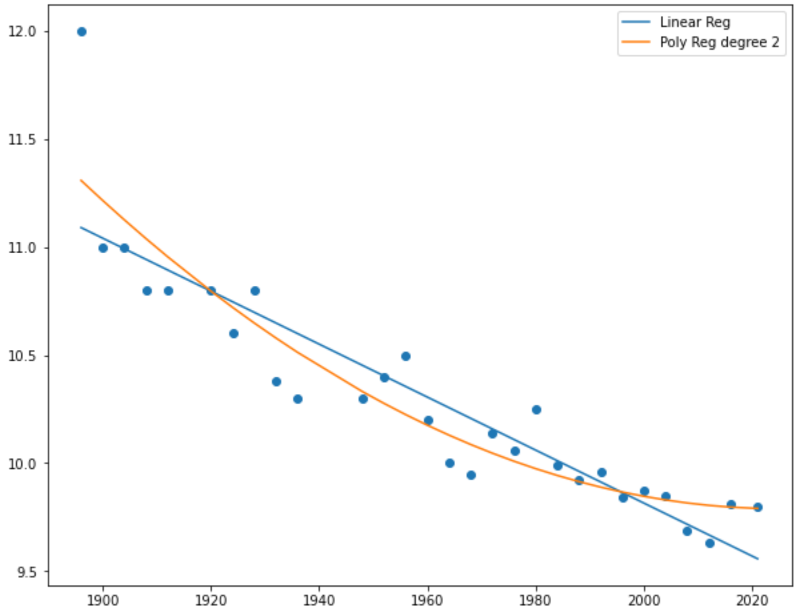
```
Out[12]: array([[1.000000e+00, 1.896000e+03, 3.594816e+06],
                [1.000000e+00, 1.900000e+03, 3.610000e+06],
                [1.000000e+00, 1.904000e+03, 3.625216e+06],
                [1.000000e+00, 1.908000e+03, 3.640464e+06],
                [1.000000e+00, 1.912000e+03, 3.655744e+06],
                [1.000000e+00, 1.920000e+03, 3.686400e+06],
                [1.000000e+00, 1.924000e+03, 3.701776e+06],
                [1.000000e+00, 1.928000e+03, 3.717184e+06],
                [1.000000e+00, 1.932000e+03, 3.732624e+06],
                [1.000000e+00, 1.936000e+03, 3.748096e+06],
                [1.000000e+00, 1.948000e+03, 3.794704e+06],
                [1.000000e+00, 1.952000e+03, 3.810304e+06],
                [1.000000e+00, 1.956000e+03, 3.825936e+06],
                [1.000000e+00, 1.960000e+03, 3.841600e+06],
                [1.000000e+00, 1.964000e+03, 3.857296e+06],
                [1.000000e+00, 1.968000e+03, 3.873024e+06],
                [1.000000e+00, 1.972000e+03, 3.888784e+06],
                [1.000000e+00, 1.976000e+03, 3.904576e+06],
                [1.000000e+00, 1.980000e+03, 3.920400e+06],
                [1.000000e+00, 1.984000e+03, 3.936256e+06],
                [1.000000e+00, 1.988000e+03, 3.952144e+06],
                [1.000000e+00, 1.992000e+03, 3.968064e+06],
                [1.000000e+00, 1.996000e+03, 3.984016e+06],
                [1.000000e+00, 2.000000e+03, 4.000000e+06],
                [1.000000e+00, 2.004000e+03, 4.016016e+06],
                [1.000000e+00, 2.008000e+03, 4.032064e+06],
                [1.000000e+00, 2.012000e+03, 4.048144e+06],
                [1.000000e+00, 2.016000e+03, 4.064256e+06],
                [1.000000e+00, 2.021000e+03, 4.084441e+06]])
```

In [13]:
```
X
```

Out[13]:
```
0     1896
1     1900
2     1904
3     1908
4     1912
5     1920
6     1924
7     1928
8     1932
9     1936
10    1948
11    1952
12    1956
13    1960
14    1964
15    1968
16    1972
17    1976
18    1980
19    1984
20    1988
21    1992
22    1996
23    2000
24    2004
25    2008
26    2012
27    2016
28    2021
Name: Year, dtype: int64
```

In [14]:
```
lr2=LinearRegression()
pr2=lr2.fit(Xp2,y)
```

In [17]:
```
# Plotting

plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
plt.legend();
```

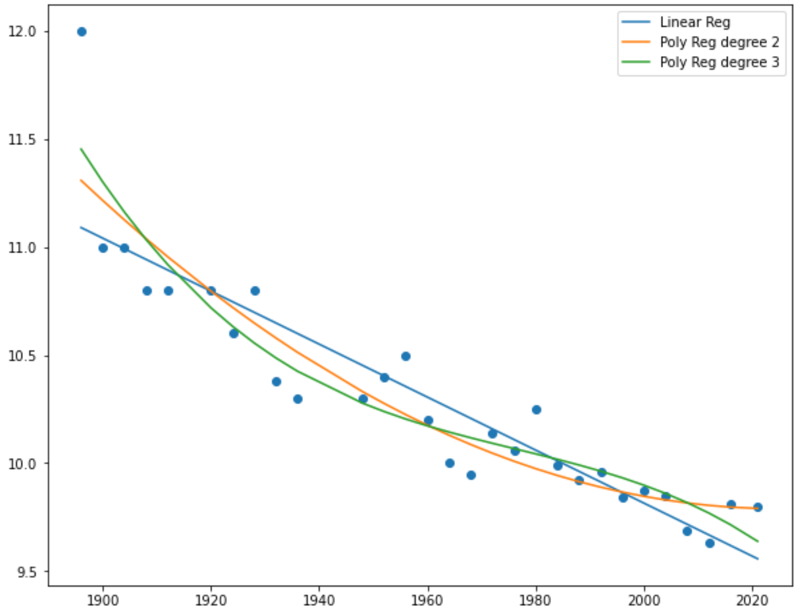


In [19]:

```
# Degree 3

p3=PolynomialFeatures(degree=3)

Xp3=p3.fit_transform(X.to_numpy().reshape(-1,1))

Xp3

lr3= LinearRegression()
pr3=lr3.fit(Xp3,y)
```

In [21]:

```
# Plotting

plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
plt.plot(X,pr3.predict(Xp3), label='Poly Reg degree 3')
plt.legend();
```

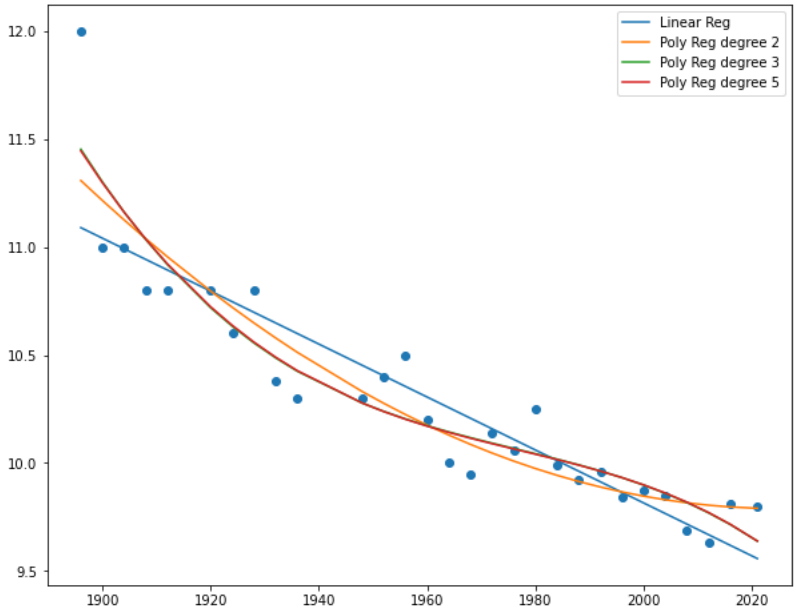


In [22]:

```
# Degree 5

p5=PolynomialFeatures(degree=5)

Xp5=p5.fit_transform(X.to_numpy().reshape(-1,1))

lr5= LinearRegression()
pr5=lr5.fit(Xp5,y)
```

In [23]:

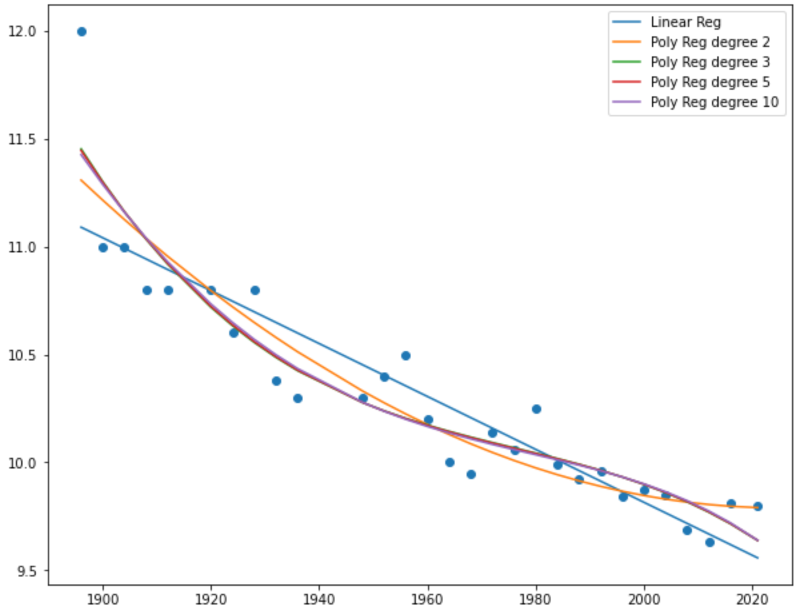
```
# Plotting

plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
plt.plot(X,pr3.predict(Xp3), label='Poly Reg degree 3')
plt.plot(X,pr5.predict(Xp5), label='Poly Reg degree 5')
plt.legend();
```



In [24]:

```
# Degree 10

p10=PolynomialFeatures(degree=10)

Xp10=p10.fit_transform(X.to_numpy().reshape(-1,1))

lr10= LinearRegression()
pr10=lr10.fit(Xp10,y)
```

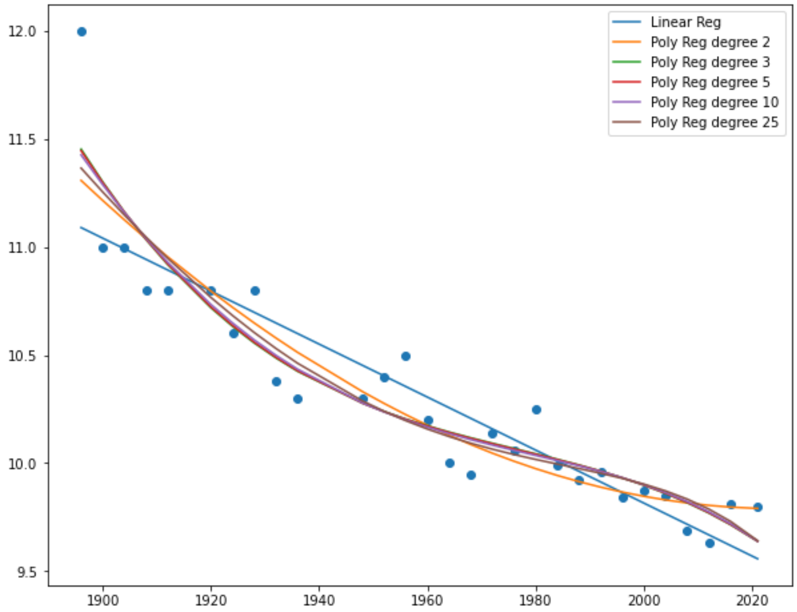
In [25]:
```
# Plotting

plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
plt.plot(X,pr3.predict(Xp3), label='Poly Reg degree 3')
plt.plot(X,pr5.predict(Xp5), label='Poly Reg degree 5')
plt.plot(X,pr10.predict(Xp10), label='Poly Reg degree 10')
plt.legend();
```



In [26]:
```
# Degree 25

p25=PolynomialFeatures(degree=25)

Xp25=p25.fit_transform(X.to_numpy().reshape(-1,1))

lr25= LinearRegression()
pr25=lr25.fit(Xp25,y)
```

In [27]:

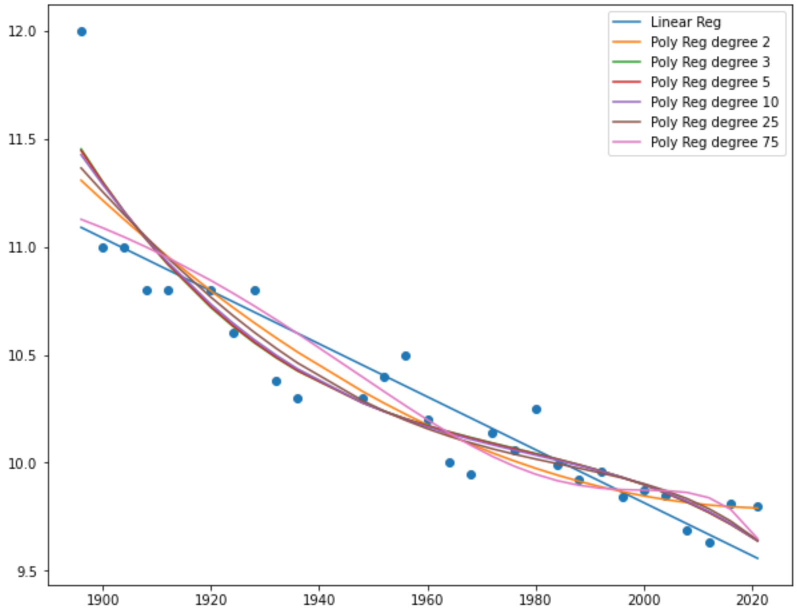
```
# Plotting

plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
plt.plot(X,pr3.predict(Xp3), label='Poly Reg degree 3')
plt.plot(X,pr5.predict(Xp5), label='Poly Reg degree 5')
plt.plot(X,pr10.predict(Xp10), label='Poly Reg degree 10')
plt.plot(X,pr25.predict(Xp25), label='Poly Reg degree 25')
plt.legend();
```



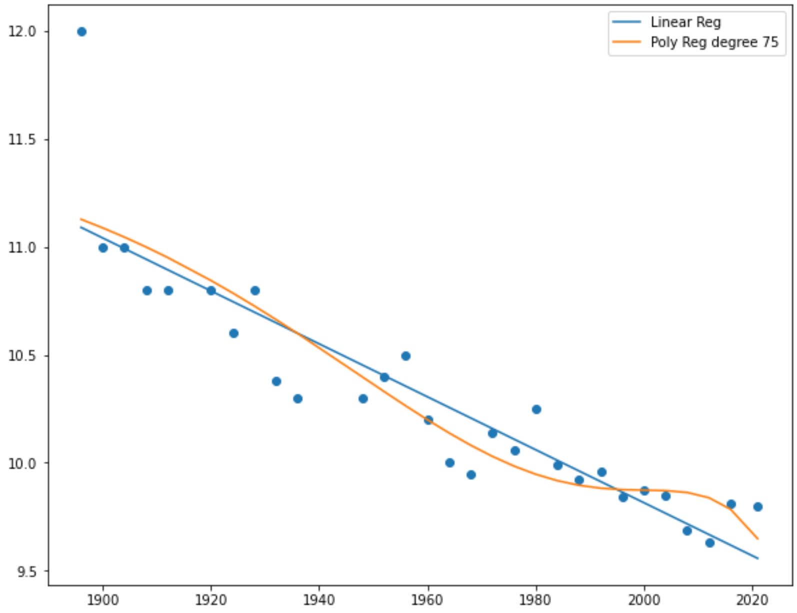
In [28]:

```
# Degree 75

p75=PolynomialFeatures(degree=75)

Xp75=p75.fit_transform(X.to_numpy().reshape(-1,1))

lr75= LinearRegression()
pr75=lr75.fit(Xp75,y)
```

In [29]:
```
plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
plt.plot(X,pr3.predict(Xp3), label='Poly Reg degree 3')
plt.plot(X,pr5.predict(Xp5), label='Poly Reg degree 5')
plt.plot(X,pr10.predict(Xp10), label='Poly Reg degree 10')
plt.plot(X,pr25.predict(Xp25), label='Poly Reg degree 25')
plt.plot(X,pr75.predict(Xp75), label='Poly Reg degree 75')
plt.legend();
```



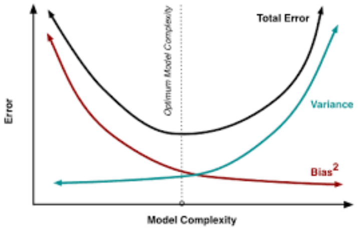
In [31]:
```
plt.figure(figsize=(10,8))
plt.scatter(X,y)
plt.plot(X,lr.predict(X.to_numpy().reshape(-1,1)),label='Linear Reg')
# plt.plot(X,pr2.predict(Xp2), label='Poly Reg degree 2' )
# plt.plot(X,pr3.predict(Xp3), label='Poly Reg degree 3')
# plt.plot(X,pr5.predict(Xp5), label='Poly Reg degree 5')
# plt.plot(X,pr10.predict(Xp10), label='Poly Reg degree 10')
# plt.plot(X,pr25.predict(Xp25), label='Poly Reg degree 25')
plt.plot(X,pr75.predict(Xp75), label='Poly Reg degree 75')
plt.legend();
```



Errors:

```
Bias error

        Corresponding the model selected >>> results in undefitting

Variance error

        Corresponding to the training data >>> results overfitting
```

Bias variance trade-off



In [ ]: