lab1

March 3, 2019

0.1 Warm up homework:

Plot how the mean square error changes with the polynomial degree ranging between 1 and 30. Do this for the cosine and some other function of your choosing. See what happens if you increase the measurment noise. author: **Maria Izabela Lewandowska**

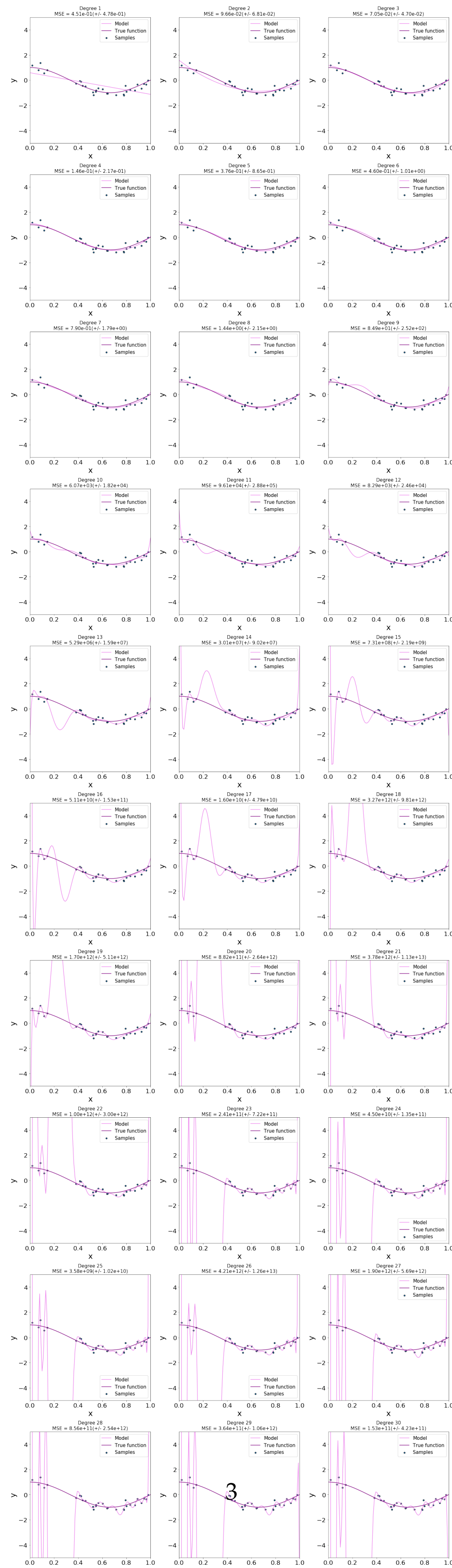
```
In [1]: import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns

        from sklearn.pipeline import Pipeline
        from sklearn.preprocessing import PolynomialFeatures
        from sklearn.linear_model import LinearRegression
        from sklearn.model_selection import cross_val_score
```

```
In [2]: # Two different True functions

        def true_fun(X):
            return np.cos(1.5 * np.pi * X)

        def true_fun3(X, a =1):
            return (np.sin(a * X)/(X+0.0001))
```

```
In [3]: # PLOT PARAMETERS FOR true_func():

        xmin = 0
        xmax = 1
        ymin = -5
        ymax = 5

        font_size = 25
        color = "purple"
        x_label = "x"
        y_label = "y"


        degrees = np.arange(1,31)
```

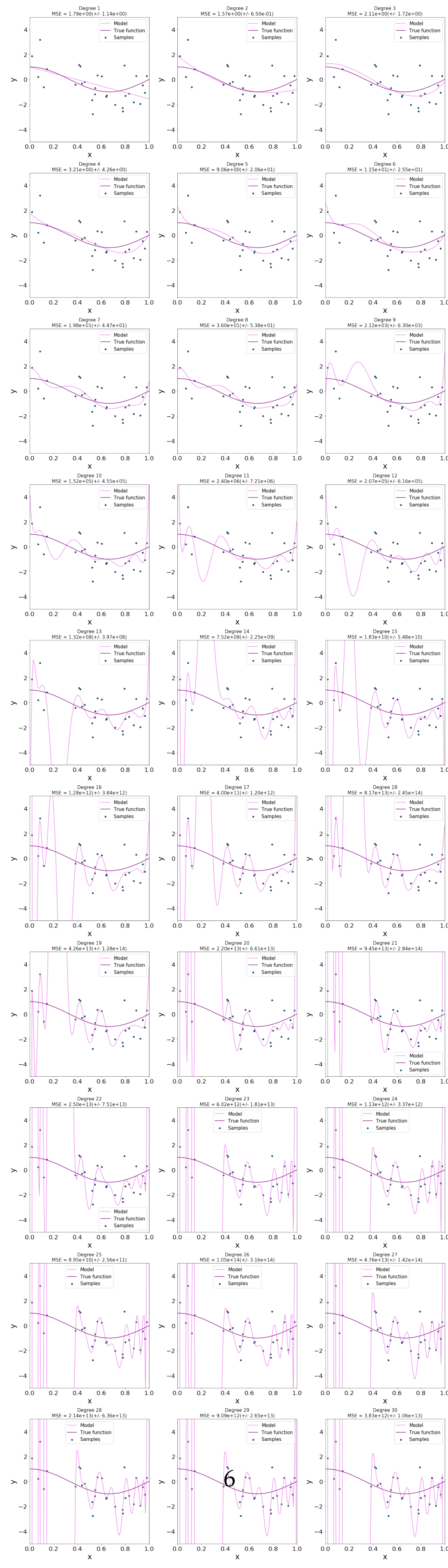
```
np.random.seed(0)

n_samples = 30

X = np.sort(np.random.rand(n_samples)) * (xmax - xmin) + xmin
y = true_fun(X) + np.random.randn(n_samples) * 0.2
```

In [4]:
```
fig, axs = plt.subplots(nrows = 10, ncols = 3,  figsize = (20,70))
i = 0
X_test = np.linspace(0, 1, 100)
statistics = []

for row in axs:
    for ax in row:

        polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False
        linear_regression = LinearRegression()

        pipeline = Pipeline([("polynomial_features", polynomial_features), ("linear_reg
        pipeline.fit(X[:, np.newaxis], y)

        # Evaluate the models using crossvalidation
        scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring="neg_mean_squa

        ax.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), color = 'violet', labe
        ax.plot(X_test, true_fun(X_test), color='purple', label="True function")
        ax.scatter(X, y, edgecolor= 'black', s=20, label="Samples")

        ax.set_ylabel(y_label)
        ax.yaxis.label.set_size( font_size )
        ax.set_ylim(ymin, ymax)
        ax.set_xlabel(x_label)
        ax.xaxis.label.set_size( font_size )
        ax.set_xlim(xmin, xmax)
        ax.tick_params(labelcolor = "black", labelsize = 20)
        ax.legend(loc="best", fontsize = 15)

        statistics.append( np.array([i+1, -scores.mean(), scores.std()]) )
        ax.set_title("Degree {:}\nMSE = {:.2e}(+/- {:.2e})".format(i+1,-scores.mean(),s
                     fontsize = 15    )
        i = i+1

fig.tight_layout()
```


Degree 1
MSE = 4.51e-01(+/- 4.78e-01)
Degree 2
MSE = 9.66e-02(+/- 6.81e-02)
Degree 3
MSE = 7.05e-02(+/- 4.70e-02)
Degree 4
MSE = 1.46e-01(+/- 2.17e-01)
Degree 5
MSE = 3.76e-01(+/- 8.65e-01)
Degree 6
MSE = 4.60e-01(+/- 1.01e+00)
Degree 7
MSE = 7.90e-01(+/- 1.79e+00)
Degree 8
MSE = 1.44e+00(+/- 2.15e+00)
Degree 9
MSE = 8.49e+01(+/- 2.52e+02)
Degree 10
MSE = 6.07e+03(+/- 1.82e+04)
Degree 11
MSE = 9.61e+04(+/- 2.88e+05)
Degree 12
MSE = 8.29e+03(+/- 2.46e+04)
Degree 13
MSE = 5.29e+06(+/- 1.59e+07)
Degree 14
MSE = 3.01e+07(+/- 9.02e+07)
Degree 15
MSE = 7.31e+08(+/- 2.19e+09)
Degree 16
MSE = 5.11e+10(+/- 1.53e+11)
Degree 17
MSE = 1.60e+10(+/- 4.79e+10)
Degree 18
MSE = 3.27e+12(+/- 9.81e+12)
Degree 19
MSE = 1.70e+12(+/- 5.11e+12)
Degree 20
MSE = 8.82e+11(+/- 2.64e+12)
Degree 21
MSE = 3.78e+12(+/- 1.13e+13)
Degree 22
MSE = 1.00e+12(+/- 3.00e+12)
Degree 23
MSE = 2.41e+11(+/- 7.22e+11)
Degree 24
MSE = 4.50e+10(+/- 1.35e+11)
Degree 25
MSE = 3.58e+09(+/- 1.02e+10)
Degree 26
MSE = 4.21e+12(+/- 1.26e+13)
Degree 27
MSE = 1.90e+12(+/- 5.69e+12)
Degree 28
MSE = 8.56e+11(+/- 2.54e+12)
Degree 29
MSE = 3.64e+11(+/- 1.06e+12)
Degree 30
MSE = 1.53e+11(+/- 4.23e+11)
Model
True function
Samples
x
y

```
In [5]: # PLOT PARAMETERS FOR true_func():

        xmin = 0
        xmax = 1
        ymin = -5
        ymax = 5

        font_size = 25
        color = "purple"
        x_label = "x"
        y_label = "y"


        degrees = np.arange(1,31)
        np.random.seed(0)

        n_samples = 30

        X = np.sort(np.random.rand(n_samples)) * (xmax - xmin) + xmin
        y = true_fun(X) + np.random.randn(n_samples) * 1

In [6]: fig, axs = plt.subplots(nrows = 10, ncols = 3,  figsize = (20,70))
        i = 0
        X_test = np.linspace(0, 1, 100)
        statistics2 = []

        for row in axs:
            for ax in row:

                polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False
                linear_regression = LinearRegression()

                pipeline = Pipeline([("polynomial_features", polynomial_features), ("linear_reg
                pipeline.fit(X[:, np.newaxis], y)

                # Evaluate the models using crossvalidation
                scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring="neg_mean_squa

                ax.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), color = 'violet', labe
                ax.plot(X_test, true_fun(X_test), color='purple', label="True function")
                ax.scatter(X, y, edgecolor= 'black', s=20, label="Samples")

                ax.set_ylabel(y_label)
                ax.yaxis.label.set_size( font_size )
                ax.set_ylim(ymin, ymax)
```

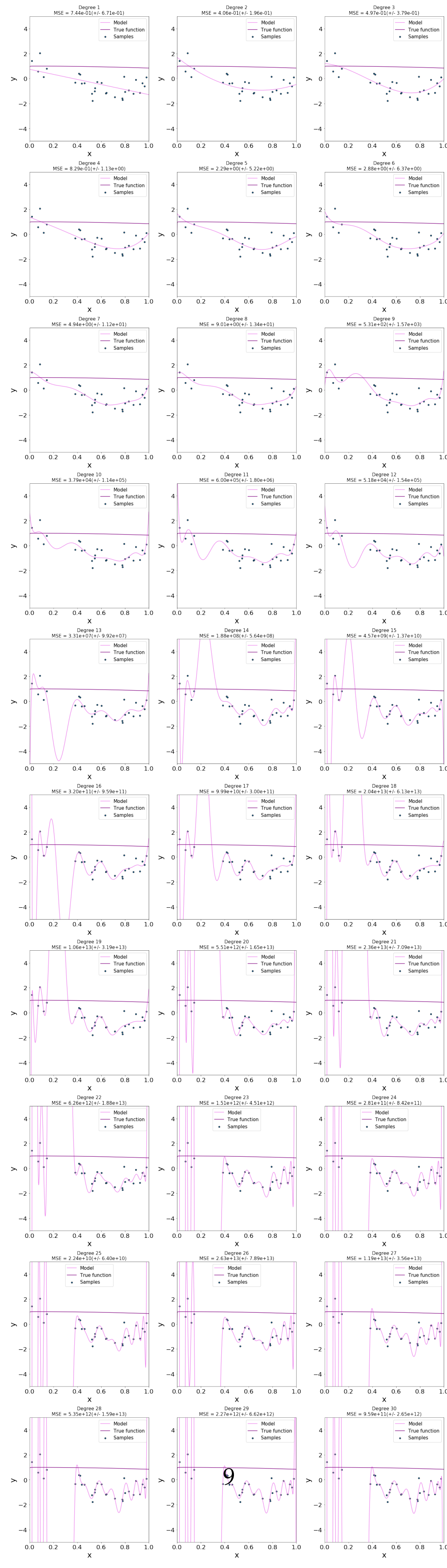
```
        ax.set_xlabel(x_label)
        ax.xaxis.label.set_size( font_size )
        ax.set_xlim(xmin, xmax)
        ax.tick_params(labelcolor = "black", labelsize = 20)
        ax.legend(loc="best", fontsize = 15)

        statistics2.append( np.array([i+1, -scores.mean(), scores.std()]) )
        ax.set_title("Degree {:}\nMSE = {:.2e}(+/- {:.2e})".format(i+1,-scores.mean(),s
                     fontsize = 15    )
        i = i+1

fig.tight_layout()
```


Degree 1
MSE = 1.79e+00(+/- 1.14e+00)
Degree 2
MSE = 1.57e+00(+/- 6.50e-01)
Degree 3
MSE = 2.11e+00(+/- 1.72e+00)
Degree 4
MSE = 3.21e+00(+/- 4.26e+00)
Degree 5
MSE = 9.06e+00(+/- 2.06e+01)
Degree 6
MSE = 1.15e+01(+/- 2.55e+01)
Degree 7
MSE = 1.98e+01(+/- 4.47e+01)
Degree 8
MSE = 3.60e+01(+/- 5.38e+01)
Degree 9
MSE = 2.12e+03(+/- 6.30e+03)
Degree 10
MSE = 1.52e+05(+/- 4.55e+05)
Degree 11
MSE = 2.40e+06(+/- 7.21e+06)
Degree 12
MSE = 2.07e+05(+/- 6.16e+05)
Degree 13
MSE = 1.32e+08(+/- 3.97e+08)
Degree 14
MSE = 7.52e+08(+/- 2.25e+09)
Degree 15
MSE = 1.83e+10(+/- 5.48e+10)
Degree 16
MSE = 1.28e+12(+/- 3.84e+12)
Degree 17
MSE = 4.00e+11(+/- 1.20e+12)
Degree 18
MSE = 8.17e+13(+/- 2.45e+14)
Degree 19
MSE = 4.26e+13(+/- 1.28e+14)
Degree 20
MSE = 2.20e+13(+/- 6.61e+13)
Degree 21
MSE = 9.45e+13(+/- 2.84e+14)
Degree 22
MSE = 2.50e+13(+/- 7.51e+13)
Degree 23
MSE = 6.02e+12(+/- 1.81e+13)
Degree 24
MSE = 1.13e+12(+/- 3.37e+12)
Degree 25
MSE = 8.95e+10(+/- 2.56e+11)
Degree 26
MSE = 1.05e+14(+/- 3.16e+14)
Degree 27
MSE = 4.76e+13(+/- 1.42e+14)
Degree 28
MSE = 2.14e+13(+/- 6.36e+13)
Degree 29
MSE = 9.09e+12(+/- 2.65e+13)
Degree 30
MSE = 3.83e+12(+/- 1.06e+13)
Model
True function
Samples
x
y

In [7]:
```
# PLOT PARAMETERS FOR true_func():

xmin = 0
xmax = 1
ymin = -5
ymax = 5

font_size = 25
color = "purple"
x_label = "x"
y_label = "y"


degrees = np.arange(1,31)
np.random.seed(0)

n_samples = 30

X = np.sort(np.random.rand(n_samples)) * (xmax - xmin) + xmin
y = true_fun(X) + np.random.randn(n_samples) * 0.5
```

In [8]:
```
fig, axs = plt.subplots(nrows = 10, ncols = 3,  figsize = (20,70))
i = 0
X_test = np.linspace(xmin, xmax, 1000)
statistics1 = []

for row in axs:
    for ax in row:

        polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False
        linear_regression = LinearRegression()

        pipeline = Pipeline([("polynomial_features", polynomial_features), ("linear_reg
        pipeline.fit(X[:, np.newaxis], y)

        # Evaluate the models using crossvalidation
        scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring="neg_mean_squa

        ax.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), color = 'violet', labe
        ax.plot(X_test, true_fun3(X_test), color='purple', label="True function")
        ax.scatter(X, y, edgecolor= 'black', s=20, label="Samples")

        ax.set_ylabel(y_label)
        ax.yaxis.label.set_size( font_size )
        ax.set_ylim(ymin, ymax)
```

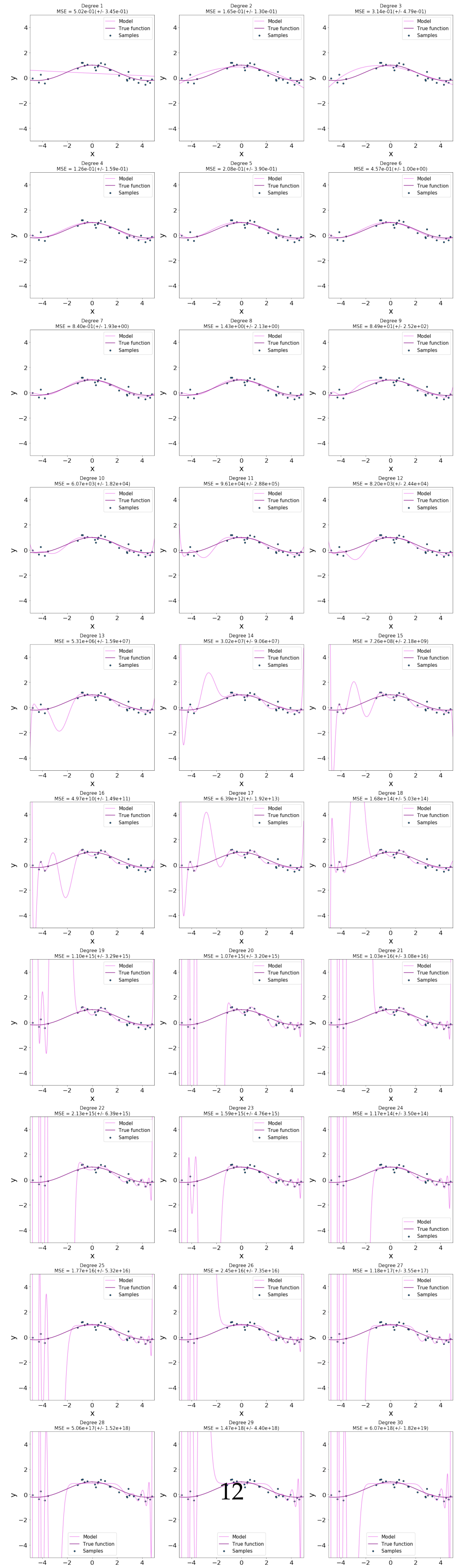
```
        ax.set_xlabel(x_label)
        ax.xaxis.label.set_size( font_size )
        ax.set_xlim(xmin, xmax)
        ax.tick_params(labelcolor = "black", labelsize = 20)
        ax.legend(loc="best", fontsize = 15)

        statistics1.append( np.array([i+1, -scores.mean(), scores.std()]) )

        ax.set_title("Degree {:}\nMSE = {:.2e}(+/- {:.2e})".format(i+1,-scores.mean(),s
                     fontsize = 15   )
        i = i+1

fig.tight_layout()
```



Degree 1
MSE = 7.44e-01(+/- 6.71e-01)
Degree 2
MSE = 4.06e-01(+/- 1.96e-01)
Degree 3
MSE = 4.97e-01(+/- 3.79e-01)
Degree 4
MSE = 8.29e-01(+/- 1.13e+00)
Degree 5
MSE = 2.29e+00(+/- 5.22e+00)
Degree 6
MSE = 2.88e+00(+/- 6.37e+00)
Degree 7
MSE = 4.94e+00(+/- 1.12e+01)
Degree 8
MSE = 9.01e+00(+/- 1.34e+01)
Degree 9
MSE = 5.31e+02(+/- 1.57e+03)
Degree 10
MSE = 3.79e+04(+/- 1.14e+05)
Degree 11
MSE = 6.00e+05(+/- 1.80e+06)
Degree 12
MSE = 5.18e+04(+/- 1.54e+05)
Degree 13
MSE = 3.31e+07(+/- 9.92e+07)
Degree 14
MSE = 1.88e+08(+/- 5.64e+08)
Degree 15
MSE = 4.57e+09(+/- 1.37e+10)
Degree 16
MSE = 3.20e+11(+/- 9.59e+11)
Degree 17
MSE = 9.99e+10(+/- 3.00e+11)
Degree 18
MSE = 2.04e+13(+/- 6.13e+13)
Degree 19
MSE = 1.06e+13(+/- 3.19e+13)
Degree 20
MSE = 5.51e+12(+/- 1.65e+13)
Degree 21
MSE = 2.36e+13(+/- 7.09e+13)
Degree 22
MSE = 6.26e+12(+/- 1.88e+13)
Degree 23
MSE = 1.51e+12(+/- 4.51e+12)
Degree 24
MSE = 2.81e+11(+/- 8.42e+11)
Degree 25
MSE = 2.24e+10(+/- 6.40e+10)
Degree 26
MSE = 2.63e+13(+/- 7.89e+13)
Degree 27
MSE = 1.19e+13(+/- 3.56e+13)
Degree 28
MSE = 5.35e+12(+/- 1.59e+13)
Degree 29
MSE = 2.27e+12(+/- 6.62e+12)
Degree 30
MSE = 9.59e+11(+/- 2.65e+12)
Model
True function
Samples
x
y

```
In [9]: # PARAMETERS:

        xmin = -5
        xmax = 5
        ymin = -5
        ymax = 5

        font_size = 25
        color = "purple"
        x_label = "x"
        y_label = "y"


        degrees = np.arange(1,31)
        np.random.seed(0)

        n_samples = 30

        X = np.sort(np.random.rand(n_samples)) * (xmax - xmin) + xmin
        y = true_fun3(X) + np.random.randn(n_samples) * 0.2

In [10]: fig, axs = plt.subplots(nrows = 10, ncols = 3,  figsize = (20,70))
         i = 0
         X_test = np.linspace(xmin, xmax, 1000)
         statistics3 = []

         for row in axs:
             for ax in row:

                 polynomial_features = PolynomialFeatures(degree=degrees[i], include_bias=False
                 linear_regression = LinearRegression()

                 pipeline = Pipeline([("polynomial_features", polynomial_features), ("linear_re
                 pipeline.fit(X[:, np.newaxis], y)

                 # Evaluate the models using crossvalidation
                 scores = cross_val_score(pipeline, X[:, np.newaxis], y, scoring="neg_mean_squa

                 ax.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), color = 'violet', lab
                 ax.plot(X_test, true_fun3(X_test), color='purple', label="True function")
                 ax.scatter(X, y, edgecolor= 'black', s=20, label="Samples")

                 ax.set_ylabel(y_label)
                 ax.yaxis.label.set_size( font_size )
                 ax.set_ylim(ymin, ymax)
```

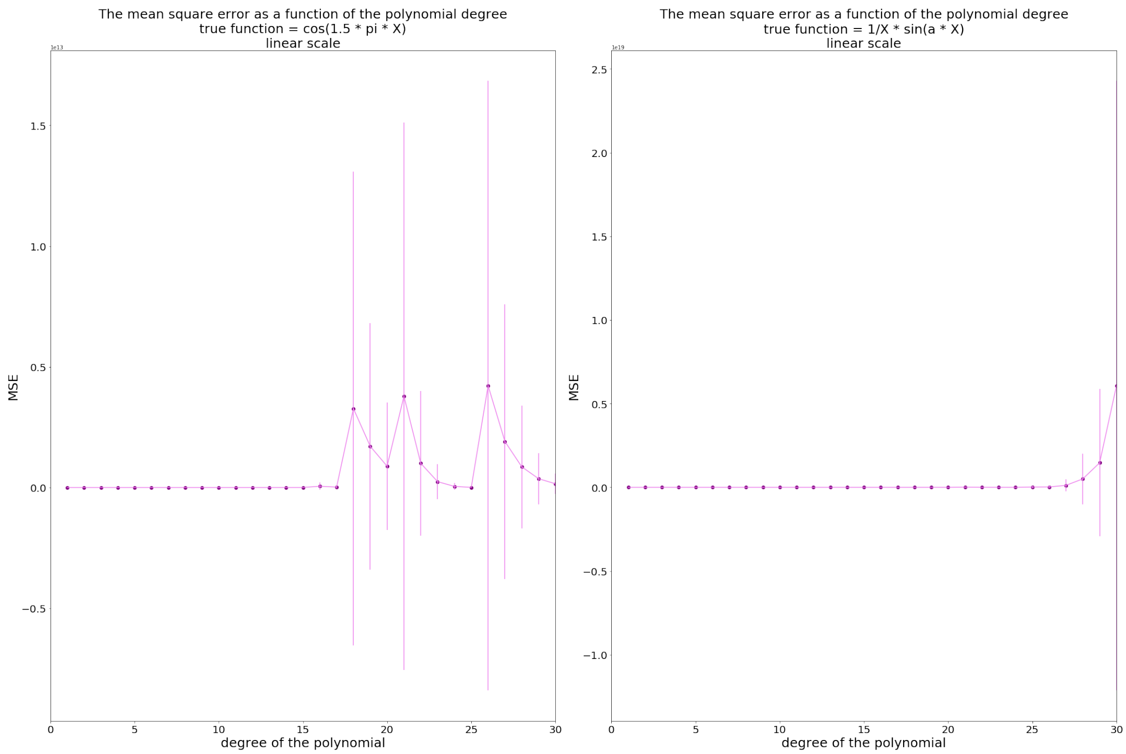
```
        ax.set_xlabel(x_label)
        ax.xaxis.label.set_size( font_size )
        ax.set_xlim(xmin, xmax)
        ax.tick_params(labelcolor = "black", labelsize = 20)
        ax.legend(loc="best", fontsize = 15)

        statistics3.append( np.array([i+1, -scores.mean(), scores.std()]) )

        ax.set_title("Degree {:}\nMSE = {:.2e}(+/- {:.2e})".format(i+1,-scores.mean()
                     fontsize = 15    )
        i = i+1

fig.tight_layout()
```


Degree 1
MSE = 5.02e-01(+/- 3.45e-01)
Degree 2
MSE = 1.65e-01(+/- 1.30e-01)
Degree 3
MSE = 3.14e-01(+/- 4.79e-01)
Degree 4
MSE = 1.26e-01(+/- 1.59e-01)
Degree 5
MSE = 2.08e-01(+/- 3.90e-01)
Degree 6
MSE = 4.57e-01(+/- 1.00e+00)
Degree 7
MSE = 8.40e-01(+/- 1.93e+00)
Degree 8
MSE = 1.43e+00(+/- 2.13e+00)
Degree 9
MSE = 8.49e+01(+/- 2.52e+02)
Degree 10
MSE = 6.07e+03(+/- 1.82e+04)
Degree 11
MSE = 9.61e+04(+/- 2.88e+05)
Degree 12
MSE = 8.20e+03(+/- 2.44e+04)
Degree 13
MSE = 5.31e+06(+/- 1.59e+07)
Degree 14
MSE = 3.02e+07(+/- 9.06e+07)
Degree 15
MSE = 7.26e+08(+/- 2.18e+09)
Degree 16
MSE = 4.97e+10(+/- 1.49e+11)
Degree 17
MSE = 6.39e+12(+/- 1.92e+13)
Degree 18
MSE = 1.68e+14(+/- 5.03e+14)
Degree 19
MSE = 1.10e+15(+/- 3.29e+15)
Degree 20
MSE = 1.07e+15(+/- 3.20e+15)
Degree 21
MSE = 1.03e+16(+/- 3.08e+16)
Degree 22
MSE = 2.13e+15(+/- 6.39e+15)
Degree 23
MSE = 1.59e+15(+/- 4.76e+15)
Degree 24
MSE = 1.17e+14(+/- 3.50e+14)
Degree 25
MSE = 1.77e+16(+/- 5.32e+16)
Degree 26
MSE = 2.45e+16(+/- 7.35e+16)
Degree 27
MSE = 1.18e+17(+/- 3.55e+17)
Degree 28
MSE = 5.06e+17(+/- 1.52e+18)
Degree 29
MSE = 1.47e+18(+/- 4.40e+18)
Degree 30
MSE = 6.07e+18(+/- 1.82e+19)
Model
True function
Samples
x
y

0.2 SUMMARY

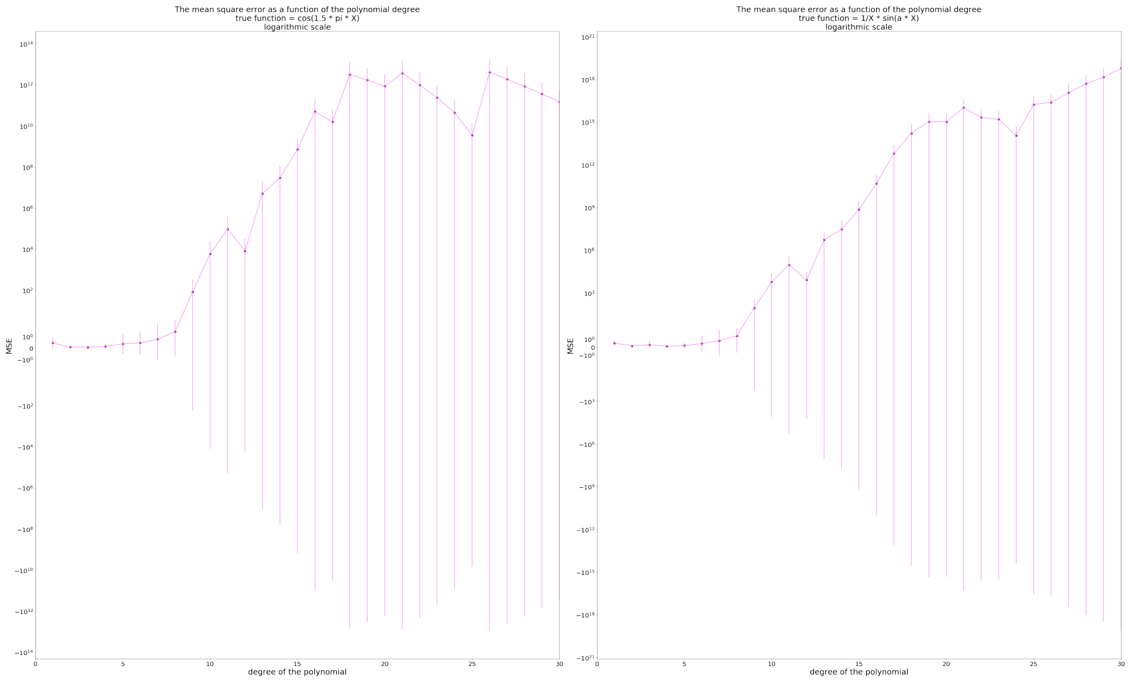
The mean square error (MSE) changes with the polynomial degree in range n ={1, 30} as shown below. For small values of the polynomial degree (n = 1) our model is underfitted. For big values (n > 8) the model is ovefitted. When the the measurment noise is increased, MSE increase too.

```
In [11]: stat = np.transpose(statistics)
         stat1 = np.transpose(statistics1)
         stat2 = np.transpose(statistics2)
         stat3 = np.transpose(statistics3)


         fig, axs = plt.subplots(nrows = 1, ncols =2,  figsize = (30,20))

         for ax in axs:
             ax.set_ylabel("MSE")
             ax.set_xlabel("degree of the polynomial")
             ax.yaxis.label.set_size( font_size )
             ax.xaxis.label.set_size( font_size )

             ax.set_xlim(0, 30)
             ax.set_xscale('linear')

             ax.tick_params(labelcolor = "black", labelsize = 20, grid_alpha = 0.5, grid_color
             #ax.grid(True)

         axs[0].errorbar(stat[0], stat[1], yerr= stat[2], color = "violet")
         axs[1].errorbar(stat3[0], stat3[1], yerr= stat3[2], color = "violet")

         axs[0].scatter(stat[0], stat[1], color = "purple")
         axs[1].scatter(stat3[0], stat3[1], color = "purple")

         axs[0].set_title("The mean square error as a function of the polynomial degree\ntrue :
         axs[1].set_title("The mean square error as a function of the polynomial degree\ntrue :

         fig.tight_layout()
```



In [12]:
```
fig, axs = plt.subplots(nrows = 1, ncols = 2,  figsize = (50,30))

for ax in axs:
    ax.set_ylabel("MSE")
    ax.set_xlabel("degree of the polynomial")
    ax.yaxis.label.set_size( font_size )
    ax.xaxis.label.set_size( font_size )

    ax.set_xlim(0, 30)
    ax.set_yscale('symlog')
    ax.set_xscale('linear')

    ax.tick_params(labelcolor = "black", labelsize = 20, grid_alpha = 0.5, grid_color
    #ax.grid(True)

axs[0].errorbar(stat[0], stat[1], yerr= stat[2], color = "violet")
axs[1].errorbar(stat3[0], stat3[1], yerr= stat3[2], color = "violet")

axs[0].scatter(stat[0], stat[1], color = "purple")
axs[1].scatter(stat3[0], stat3[1], color = "purple")

axs[0].set_title("The mean square error as a function of the polynomial degree\ntrue 
axs[1].set_title("The mean square error as a function of the polynomial degree\ntrue 
```

```
fig.tight_layout()
```




```
In [13]: fig, axs = plt.subplots(nrows = 1, ncols = 3,  figsize = (50,30))

         for ax in axs:
             ax.set_ylabel("MSE")
             ax.set_xlabel("degree of the polynomial")
             ax.yaxis.label.set_size( font_size )
             ax.xaxis.label.set_size( font_size )

             ax.set_xlim(0, 30)
             #ax.set_yscale('symlog')
             ax.set_xscale('linear')

             ax.tick_params(labelcolor = "black", labelsize = 20, grid_alpha = 0.5, grid_color
             #ax.grid(True)

         axs[0].errorbar(stat[0], stat[1], yerr= stat[2], color = "violet")
         axs[1].errorbar(stat1[0], stat1[1], yerr= stat1[2], color = "violet")
         axs[2].errorbar(stat2[0], stat2[1], yerr= stat2[2], color = "violet")

         axs[0].scatter(stat[0], stat[1], color = "purple")
         axs[1].scatter(stat1[0], stat1[1], color = "purple")
         axs[2].scatter(stat2[0], stat2[1], color = "purple")
```

```python
axs[0].set_title("The mean square error as a function of the polynomial degree\ntrue :
axs[1].set_title("The mean square error as a function of the polynomial degree\ntrue :
axs[2].set_title("The mean square error as a function of the polynomial degree\ntrue :


fig.tight_layout()
```