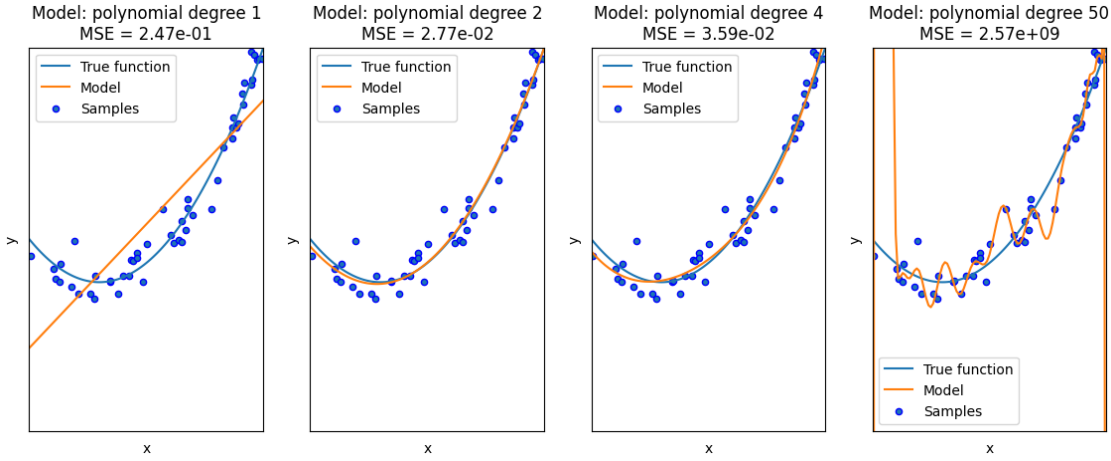
# Under_Over_Fitting

January 13, 2023

[1]:
```
# Adapted from:
# https://scikit-learn.org/stable/auto_examples/model_selection/
 ↪plot_underfitting_overfitting.html

# Install the required packages, and suppress any warnings if they are already␣
 ↪installed.
import sys
!{sys.executable} -m pip install numpy matplotlib scikit-learn | grep -v␣
 ↪'already satisfied'

import numpy as np
import matplotlib.pyplot as plt

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

# Uncomment if we want the same results every time.
#np.random.seed(0)

# True function is a quadratic.
def true_function(x):
    return (5 *  x * x - 3 * x)

number_of_samples = 50

# The degrees of polynomial to fit.
polynomial_degrees = [1, 2, 4, 50]

# Randomly selected x values in 0-1.
x_values = np.sort(np.random.rand(number_of_samples))

noise_factor = 0.2

# Normally distributed noise.
noise = np.random.randn(number_of_samples) * noise_factor
```

```
y_values = true_function(x_values) + noise

plt.figure(figsize=(14, 5))

for i in range(len(polynomial_degrees)):

    polynomial_features = PolynomialFeatures(degree=polynomial_degrees[i])
    linear_regression = LinearRegression()
    pipeline = Pipeline(
        [
            ("polynomial_features", polynomial_features),
            ("linear_regression", linear_regression),
        ]
    )

    x_values_as_column_vector = x_values[:, None]
    pipeline.fit(x_values_as_column_vector, y_values)

    # Evaluate the models using cross validation.
    scores = cross_val_score(
        pipeline, x_values_as_column_vector, y_values,␣
 ↪scoring="neg_mean_squared_error", cv=10
    )

    # Regular plotting positions at 0.01 intervals.
    x_plot_values = np.linspace(0, 1, 101)
    x_plot_values_as_column_vector = x_plot_values[:, None]

    true_y_values = true_function(x_plot_values)
    predicted_y_values = pipeline.predict(x_plot_values_as_column_vector)

    axes = plt.subplot(1, len(polynomial_degrees), i + 1)

    plt.setp(axes, xticks=(), yticks=())
    plt.plot(x_plot_values, true_y_values, label="True function")
    plt.plot(x_plot_values, predicted_y_values, label="Model")
    plt.scatter(x_values, y_values, edgecolor="b", s=20, label="Samples")
    plt.xlabel("x")
    plt.ylabel("y")
    plt.xlim((0, 1))
    plt.ylim((-2, 2))
    plt.legend(loc="best")
    plt.title("Model: polynomial degree {}\nMSE = {:.2e}".
 ↪format(polynomial_degrees[i], -scores.mean()))

plt.show()
```


Model: polynomial degree 1
MSE = 2.47e-01
Model: polynomial degree 2
MSE = 2.77e-02
Model: polynomial degree 4
MSE = 3.59e-02
Model: polynomial degree 50
MSE = 2.57e+09
True function
Model
Samples
x
y