**6) OVERFITTING/UNDERFITTING:**

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

def true\_fun(X):

    return np.cos(1.5 \* np.pi \* X)

np.random.seed(0)

n\_samples = 30

degrees = [1, 4, 15]

X = np.sort(np.random.rand(n\_samples))

y = true\_fun(X) + np.random.randn(n\_samples) \* 0.1

plt.figure(figsize=(14, 5))

for i in range(len(degrees)):

    ax = plt.subplot(1, len(degrees), i + 1)

    plt.setp(ax, xticks=(), yticks=())

    polynomial\_features = PolynomialFeatures(degree=degrees[i], include\_bias=False)

    linear\_regression = LinearRegression()

    pipeline = Pipeline(

        [

            ("polynomial\_features", polynomial\_features),

            ("linear\_regression", linear\_regression),

        ]

    )

    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation

    scores = cross\_val\_score(

        pipeline, X[:, np.newaxis], y, scoring="neg\_mean\_squared\_error", cv=10

    )

    X\_test = np.linspace(0, 1, 100)

    plt.plot(X\_test, pipeline.predict(X\_test[:, np.newaxis]), label="Model")

    plt.plot(X\_test, true\_fun(X\_test), label="True function")

    plt.scatter(X, y, 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(

        "Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(

            degrees[i], -scores.mean(), scores.std()

        )

    )

plt.show()

output:

