Report of the 2nd project

Yasmin Bouhdada
Annalisa Dettori
Andrea Schinoppi

1 Introduction

In this project we aim to apply and evaluate the performance of some regression methods such as kNN regression, dummy regressor, linear regression and polynomial regression. These methods will be applied to four data sets, two of them are artificially made while two are real data sets regarding the problem of house pricing. To evaluate the performance of these methods we will use the MSE and R^2 often together with cross validation. We will also compare the data with the different approaches.

2 Data Set

2.1 First data set

The first data set was made of one column X for the independent variable and one column for the dependent variable y. The data set had 300 samples and it has no NaNs. We didn't know what the variables represented, but we could resume the main information of the variables with this table:

	mean	std dev	min	max
X	12.39	7.04	0.07	25.12
y	−0.03	0.73	−1.41	1.29

2.2 Second data set

The second data set was made of two columns X_1, X_2 for the independent variables and one column for the dependent variable y. The data set had 500 samples and it had no NaNs. We didn't know what the variables represented, but we could resume the main information of the variables with this table:

	X_1	X_2	y
mean	0.06	−0.36	−2.42
std dev	0.06	0.37	−2.42
min	−7.14	−0.73	−52.57
max	0.12	0.01	−52.57

2.3 Third data set

The third data set (named *real world data*) was a real data set regarding the prices of some houses. It was made of ten columns $X_1, \ldots, X_{10}$ for the independent variables and one column for the dependent variable y. The data set had 1095 samples and it had no NaNs.

The labels of the variables were: "LotArea", "TotalBsmtSF", "1stFlrSF", "2ndFlrSF", "GrLivArea", "WoodDeckSF", "OpenPorchSF", "3SsnPoarch", "ScreenPorch" and "PoolArea".

We could resume the main information of the variables with this table:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
mean	10722.41	1159.84	0	338.71	1505.13	91, 06	47.26	2.78	15.09	2.14
std dev	11054.40	376.46	34900	432.04	514.24	120.64	66.79	25.18	56.55	35.79
min	1300	0	343	0	334	0	0	0	0	0
max	215245	3206	3228	1872	4676	670	547	407	480	738

2.4 Fourth data set

The fourth data set (named *new data*) set was an additional part to the previous one, so it still dealed with the prices of the houses. It was made of ten columns $X_1, \ldots, X_{10}$ for the independent variables and one column for the dependent variable y. The data set had 365 samples and it had 100 NaNs in the column y.

The labels of the variables are: "LotArea", "TotalBsmtSF", "1stFlrSF", "2ndFlrSF", "GrLivArea", "WoodDeckSF", "OpenPorchSF", "3SsnPoarch", "ScreenPorch" and "PoolArea".

The two approaches we used to fill the NaNs were:

- set the missing values as the feature mean, which provided the following results:

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	X_8	X_9	X_{10}
mean	9900.07	1061.60	1170.99	371.83	1560.45	103.80	44.85	5.30	14.99	4.61
std dev	5578.82	496.67	414.90	448.22	483.72	137.89	64.50	39.10	53.23	51.03
min	1491	0	483	0	630	0	0	0	0	0
max	63887.00	6110	4692	2065	5642	857	418	508	322	648

- derive the missing values from the neighbours, which provided the same results.

We had to take into account that the proportion of NaNs was very high very high, so the statistical methods might not be that accurate.

3 Foundations

3.1 Methods

3.1.1 KNN classifier for regression

The first method we implemented was kNN classifier for regression by creating a class with three objects: `init`, `fit` and `predict`. To provide the predictions, we created a neighbourhood of k samples for each sample as we did in classification and, instead of returning the class owned by most neighbours, we returned the mean on the dependent

variable y of the k closest samples throught `distances = np.abs(self.X - x)`. In such a way we predicted for a new sample the mean value of the k closest samples.

After using it for the first dataset we chose to use `KNeighborsRegressor` that we imported from `sklearn.neighbors`.

3.1.2 Dummy regressor

Another method we implemented by classes was the dummy regressor for which we created the objects `init`, `fit` and `predict`. The dummy regressor only provides as prediction the mean of the dependent variable y: `self.prediction_value = np.mean(y)`

3.1.3 Linear regression

From `sklearn.linear_model` we imported `LinearRegression` without implemeting anything.

3.1.4 Polynomial regression

From `sklearn.processing` we imported `PolynomialFeatures` without implemeting anything.

3.2 Evaluation

3.2.1 Cross validation

For kNN regression for the first data set we also implemented the cross validation for which we splitted the data set into 5 folds. To do it we trained the model on four folds and we tested on the fifth one to compute the MSE (or the R^2). After doing it for all folds, we took the mean of the five MSE (or of the five R^2) obtained.

From the second data set and so on, we used the `cross_val_score` imported from *sklearn.model_selection*

3.2.2 MSE and R^2

For the first data set we evaluated the MSE, whereas for the other data sets we evaluated the R^2. We wanted the MSE to be as close as possible to zero since it represents the quantity of error we are making from the real values predicted. On the other hand, since the $R^2 \in [0, 1]$, we wanted the R^2 to be as close as possible to 1, where 1 means perfect predictions.

4 Experiments

4.1 Set-Up

Our experiments were carried on a jupyter notebook that we converted in a python file. The data sets' extension is `.npz`.

4.2 Analysis and Results

4.2.1 First data set

In the first data set the results of the MSE we obtained through the cross validation were:

	kNN	dummy regressor
MSE	0.56	0.54

We can infer that kNN for regression and the dummy regressor work both very well, because we want the MSE to be close to zero and in fact it is very small.

We also tested the model for different values of $k \in [2, 100]$ (without the cross validation) to observe a good generalization, an over fitting and an under fitting. What we obtained is that:

- we maximize the MSE for $k = 89$, so we have an under fitting;
- we minimize the MSE for $k = 6$, so we have an over fitting;
- we have a good generalization for $k = 51$.

We can compare the data obtained in the following table:

	$k = 6$	$k = 51$	$k = 89$
MSE	0.04	0.43	0.67

4.2.2 Second data set, non - standardized data

For the second data set we trained three models: a linear regression model, with a polynomial regression model of degree 2 and with a kNN for regression model with $k = 5$. For all of them we computed the R^2 through a 5-fold cross validation. The results obtained are the following:

	linear regression	polynomial regression	kNN
R^2	0.31	0.96	0.12

As we can see kNN regression with $k = 5$ returns the lowest value for the R^2, synthom that it's the worst model to use for this data set.

For linear regressione the value of the R^2 is still small, so we can't also trust this method.

At the end for polynomial regression with degree 2 we get the biggest value for the R^2, this means that polynomial regression works very good, because it's close to 1.

By the way, what we should underline is the fact that we did not tune the parameters of the degree of polynomial regression and the number of neighbours of kNN.

4.2.3 Second data set, standardized data

We did the same computation for the standardized data and we can resume as it follows:

	linear regression	polynomial regression	kNN
R^2	0.31	0.96	0.88

Now kNN regression with $k = 5$ (wich previoulsy returned the lowest value for the R^2) provides a very high value for the R^2. Probably this is due to the fact that the data need to be comparable before applying a statistical method.

For linear regressione the value of the R^2 remains still small, so even if we standardize the data we don't get good results.

For polynomial regression with degree 2, we get the biggest value for the R^2 and if we confront the standardize data with the ones non standardized, the R^2 remains exactly the same. This may be because the two independent variables and the dependent variable have a quadratic proportion.

4.2.4 Third and fourth data sets

In the third data set we computed the R^2 of a linear regression model and on a kNN regression model with 5-fold cross validation. The results obtained are the following:

	linear regression	kNN
R^2	0.55	0.37

Both of the results are very low.

Now we take into account the fourth data set. Considering to fill the NaNs with the mean or with the neighbour samples we get the following results:

	linear regression	kNN
R^2 with the mean	0.60	0.25
R^2 with the neighbourhood	0.59	0.23

So the results doesn't change that much with these approaches.

5 Discussion

To sum up, we applied on the four data set given the statistical methods kNN regression, dummy regressor, linear regression and polynomial regression. We implemented the code for knn regression, for the dummy classifier and for the cross validation. We used the libraries to import the linear regression and the polynomial. To evaluate the performance of these methods we will used the MSE and R^2 together with cross validation and compared all the results obtained.

A Appendix

A.1 Images

1. The following image represents the distribuition of the samples of the first data set

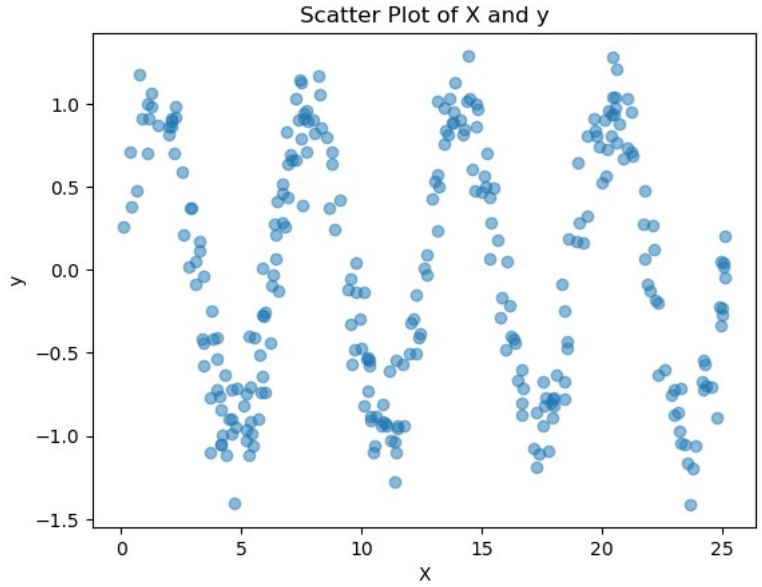

Figure A.1: Relation between MSE and k

2. The following image represents the relationship between the mean squared error and the dimension of the neighbourhood we are taking for kNN regression for the first data set.

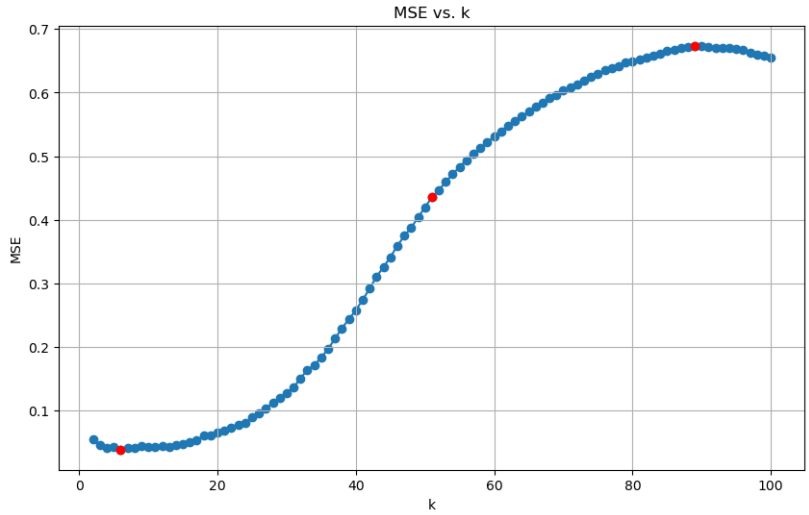

Figure A.2: Relation between MSE and k

3. The following images represents the learning curves of the data set two for linear

regression, polynomial regression and kNN regression.

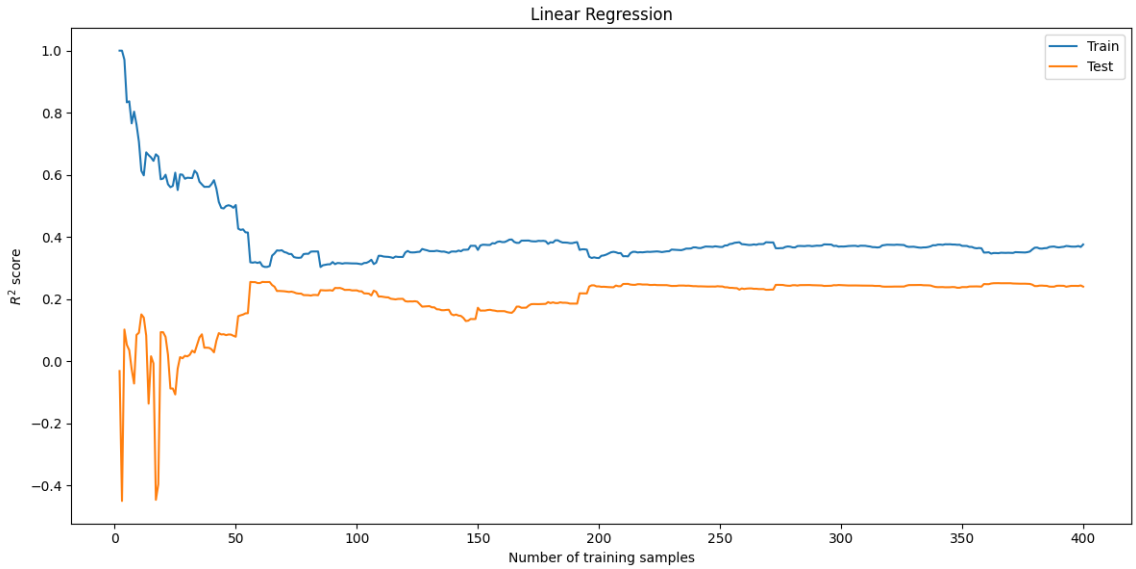

Figure A.3: Learning curve of linear regression for the second data set

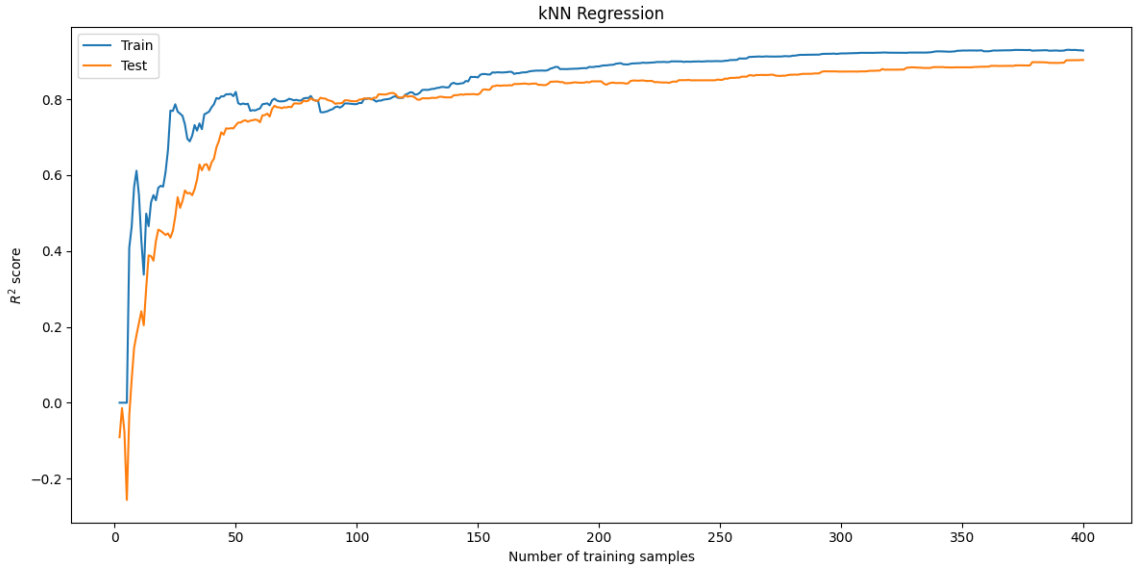

Figure A.4: Learning curve of kNN regression for the second data set

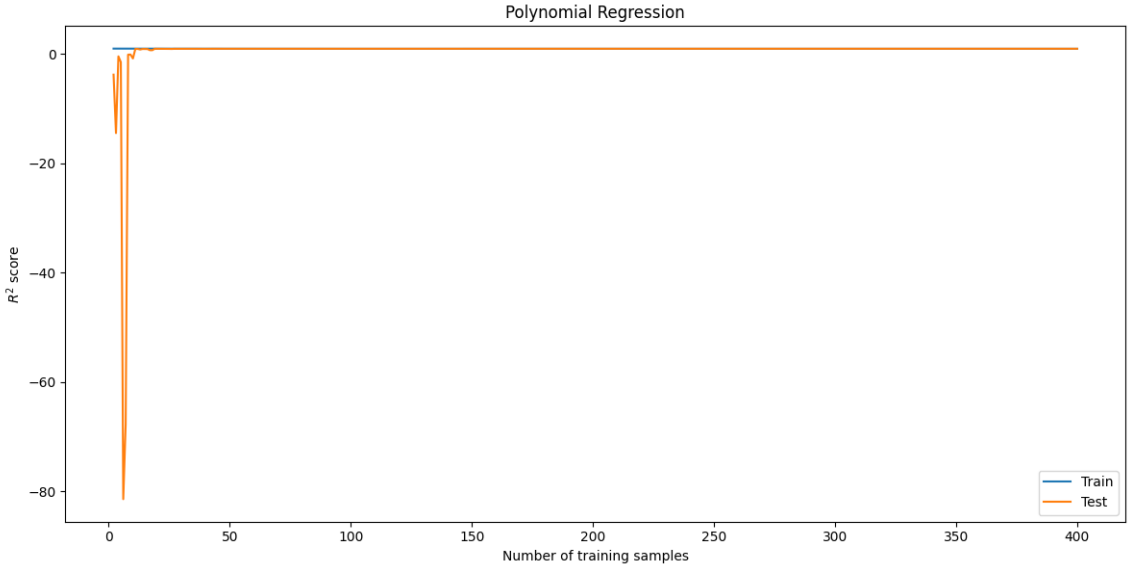

Figure A.5: Learning curve of polynomial regression for the second data set

4. The following image represents the learning curve of a linear regression model on the third data set.

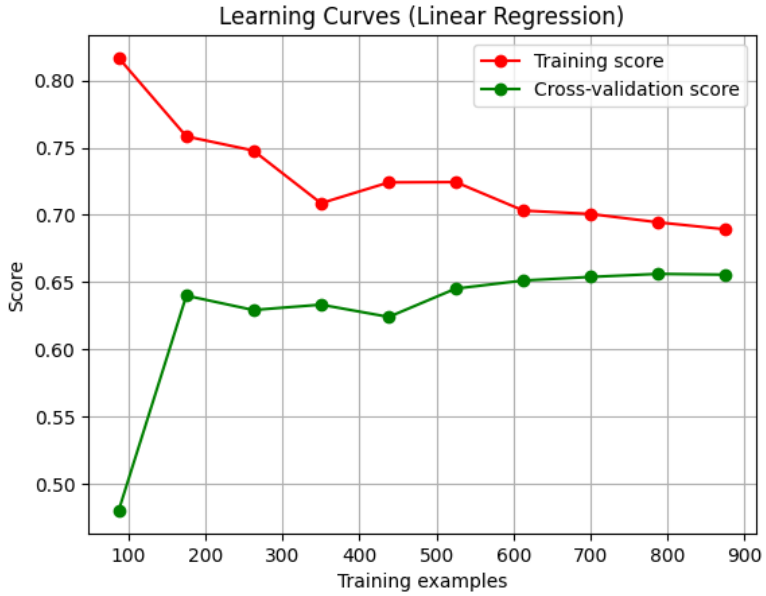

Figure A.6: Learning curve of linear regression for the third data set

5. The following image represents the learning curve of a kNN regression model on the third data set.

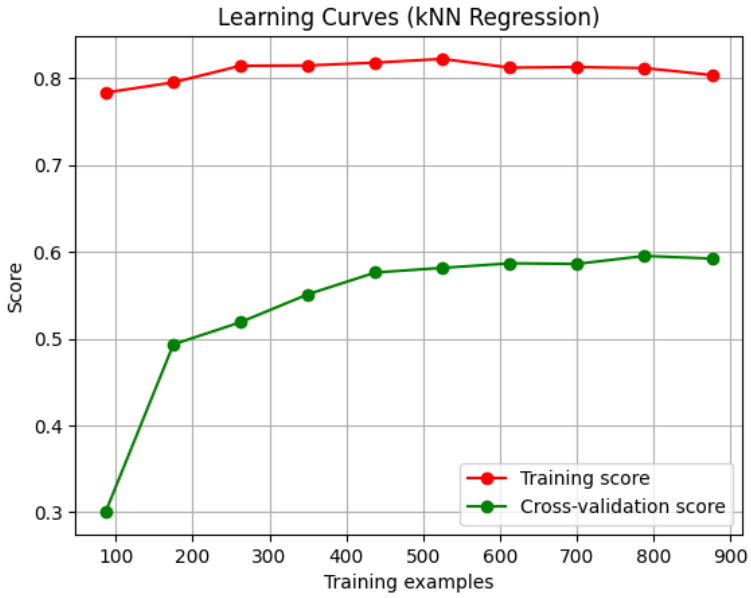

Figure A.7: Learning curve of kNN regression for the third data set

A.2 Proof of the implementation *1b*

```
class CustomKNNRegressor:
def __init__(self, k=5):
    self.k = k
    self.X = None
    self.y = None

def fit(self, X, y):
    self.X = X.reshape(-1, 1)
    self.y = y

def predict(self, X_test):
    X_test = X_test.reshape(-1, 1)
    predictions = []
    for x in X_test:
        distances = np.abs(self.X - x)
        indices = np.argsort(distances.flatten())[:self.k]
        k_nearest_y = self.y[indices]
        prediction = np.mean(k_nearest_y)
        predictions.append(prediction)
    return np.array(predictions)
```