# Data Science and Visualization, S2026

Supervised learning 1: Regression

2026-03-17

## Exercise 1: Importing and loading

In this exercise set, you will use the the scikit-learn library to fit linear regression and $K$-nearest neighbors (KNN) regression models to a dataset of taxi trips in New York City. The dataset contains the following columns:

- **pickup**: Date and time of the beginning of the trip
- **dropoff**: Date and time of end of the trip
- **passengers**: The number of passengers in the taxis
- **distance**: The distance of the trip (miles)
- **total**: The total amount charged (USD)
- **color**: The color of the taxi (yellow, green)
- **payment**: The payment method (credit card, cash)
- **pickup_zone**: The zone where the passengers were picked up
- **dropoff_zone**: The zone where the passenger were dropped off
- **pickup_borough**: The boroughs where the passengers were picked up
- **dropoff_borough**: The borough where the passengers were dropped off

We will estimate a series a model with the purpose of predicting the total fare of taxi tips. So throughout the exercise set, we will use the column **total** as the **target**. The exercise set will also ask you to visualize aspects of the data or your models. As always, make sure your plots abide to the principles of data visualization. One approach to controlling the layout of your plots is to specify a global theme with the `theme()` function. You can then modify the theme for each plot you create as needed

Use the empty code cell below to first install the scikit-learn library. Then clear the cell and use it to import the pandas and Plotnine libraries and the LinearRegression, StandardScaler KNeighborsRegressor, and submodules from the scikit-learn library. Download the file *taxis.csv* from Moodle and save it in the same folder as this notebook. Then read it as DataFrame object and assign it to the variable `df`. Finally, use a DataFrame method to find out how many rows and columns the dataset has and print the results

```
# Import libraries

# Load the taxis data

# How many rows and columns does df have?
```

```
# Specify a global theme to use in all plots
```

## Exercise 2: Handling missing values

Before we can conduct our supervised learning analysis, we will have to go through a series of **preprocessing** and **feature engineering** steps. In the first of these steps, we will detect and remove missing values

**2.1** We will start by getting an overview of the number missing values in our data. To do so, create a plot that displays the proportion of missing values on each column in `df`. You can do this by following these 3 steps:

1. Start by creating a new DataFrame where the first column contains all the column names of `df` and the second and third columns contain the percentage of missing and non-missing values in a given column. Than reshape the new DataFrame such that there are now two rows oer column of `df` and the percentages of missing and non-missing values are contained in a single column. A third column should identify whether the percentage value in a given row is for missing or non-missing values. Do this using the .melt() method
2. Order column names by percentage of missing values and remove percentages equal to zero to increase readability of the plot

3. Create the plot with Plotnine. Use the function `geom_col()`, display column names on the vertical axis, the percentages on the horizontal axis, and map the identifyer column to the `fill` aesthetic. Select colors to use for missing and non-missing values with the function `scale_fill_manual()`. Label each slice of the bars with the percentage it represents

```
# Step 1: Construct a DataFrame containing the data to be plotted
```

```
# Step 2: Reorder categorical variables and replace 0 with none
```

```
# Step 3: Create the plot
```

**2.2** Use the empty code cell below to remove all rows in `df` that have any missing values. You can use the pandas method `.dropna()` to do this. Make sure to reset the row index of `df`

```
# Remove rows with missing values
```

## Exercise 3: Feature construction

**3.1** Use the column pickup in `df` to construct a new qualitative feature indicating whether the trip started during nighttime (12 p.m. - 6 a.m.), in the morning (6 a.m. - 12 a.m.), in the afternoon (12 a.m. - 6 p.m.), or in the evening (6 p.m. - 12 p.m.). To do this, you can combine the pandas function pd.to_datemite(), the methods .dt and .hour with the function `pd.cut()`. Name the column pickup_time

```
# Construct a qualitative feature containing pickup time
```

**3.2** Use the column pickup in `df` to construct a new qualitative feature indicating which day of the week the trip started. Format the days of the week as the strings 'monday' through 'sunday'. To do this, you can combine the pandas function pd.to_datemite(), the methods .dt and .weekday with the method `.map()` taking as its argument a `lambda` function that uses a dictionary to map the integers from 0 to 6 to the days of the. Name the column pickup_day

```
# Construct a qualitative feature containing the day of the week where the trip started
```

**3.3** Use the columns pickup and dropoff in `df` to construct a new column containing the travel time of each taxi trip **in minutes**. You will need to use the pandas function pd.to_datemite() and the methods .dt and .total_seconds() to do this. Name the column travel_time

```
# Construct a new feature containing travel time in minutes
```

**3.4** Create dummy variables from the qualitative features passengers, color, pickup_borough, dropoff_borough, pickup_time, and pickup_day. Leave out the first category on each qualitative, i.e., if there are $K$ classes on a qualitative feature, construct dummies for the $2^{nd}$ through the $K^{th}$ class. Append the dummy variables to `df` as new columns

```
# Create dummies from the qualitative features listed
```

## Exercise 4: Visualizing associations

Before we specify a linear regression model, let's use scatterplots to investigate the relationships between the target total and the quantitative features distance and travel_time. You can do this by following these two steps:

1. Start by creating a DataFrame containing the data to be plotted. You will need to construct a DataFrame where the values of the quantitative features distance and travel_time are contained in a single column. A second column should identify which feature the value in a particular row pertains to. A third column should then map the value of the target to the value of the feature displayed in a particular row. You can use pandas method `.melt()` to reshape and filter `df` to have this structure
2. Create a scatterplot of the target and each of the features. Map each of the features to small multiples with the function `facet_wrap()`. Use the function `geom_point()` to plot the data. Assign the values of each of the features to the horizontal axis of the coordinate system and the values of the target to the vertical axis. Draw a LOESS curve through the cloud of points with the function `geom_smooth()` to visualize the trend in the data

```
# Step 1: Create a DataFrame containing the data to be plotted
```

```
# Step 2: Create the plot
```

## Exercise 5: Linear regression

**5.1** Use the empty code cells below to do the following:

1. Fit a multiple linear regression model of the target total on the quantitative features distance and travel_time. Put the coefficients in a new DataFrame and output it. Compute the RSE and the $R^2$ and print them. Take a minute to think about what estimated coefficients and the goodness of fit statistics tell you about the data
2. Create a **residual plot**. This is a scatterplot with the predicted values on the horizontal axis and the residuals on the vertical axis. If our linear regression approximates the true $f$ well, the residual plot should show no discernable pattern. One approach to visually inspecting this is to draw a LOESS curve through the cloud of points. You can read James et al. (2023): ISL pp. 100-101 to learn more about residual plots

```
# Fit a multiple linear regression model and output the coefficients
```

```
# Compute goodness of fit metrics
```

```
# Create a residual plot
```

**5.2** Use the empty code cells below to do the following:

1. Incorporate the qualitative features passengers, color, pickup_borough, dropoff_borough, pickup_time, and pickup_day in the multiple linear regression that you estimated in exercise 5.1 by including the dummy variables that you constructed in exercise 3.4. Put the coefficients in a new DataFrame and output it. Compute the RSE and the R^2 and print them. Take a minute to think about what estimated coefficients and the goodness of fit statistics tell you about the data
2. Create a residual plot for the model fit

```
# Fit a multiple linear regression model and output the coefficients
```

```
# Compute goodness of fit metrics
```

```
# Create a residual plot
```

## Exercise 6: $K$-nearest neighbors

Use the empty code cell to do the following:

1. Fit KNN regressions of the target total on the quantitative features distance, travel_time, color, pickup_borough, dropoff_borough, pickup_time, and pickup_day for the following values of $K$: 1, 5, 10, 20, 40, 80, 160. Standardize the features before fitting the models
2. Create a new DataFrame in which the first column contains the value of $K$ for each fit, the second column contains the RMSE, and the third column contains the $R^2$ and output it
3. Compare the goodness of fit statistics for the KNN regression with the goodness of fit statistics for the OLS regression model that you computed in exercises 5.1. Take also a moment to think about why the KNN and OLS goodness of fit statistics for the taxi trips data are so different from the same statistics for the exoplanets data presented in the lecture. What explains the relatively good performance of OLS on the taxi trips data?

```
# Fit KNN regressions for multiple values of K, compute the RMSE and the R^2, and display the resul
```

## Exercise 7: Logarithmic transformations

Aside from polynomial transformations, another very common feature transformation technique is to compute the (natural) **logarithm of a feature or the target**. In this exercise, we will return to the exoplanets data to see if we can improve on the multiple linear regression models presented in the lecture by using logarithmic transformations. For an overview of the mathematics of logarithmic transformations, go to this Wikipedia page

**7.1** Use the empty code cell below to do the following:

1. Download the file *exoplanets.csv* from Moodle and save it in the same folder as this notebook. Then read it as DataFrame object and assign it to the variable `df_1`
2. Use the column disc_facility and the the function `pd.get_dummies()` to construct dummy variables for discovery facilities. Leave out the category 'Other facility'. We will use this category as the reference category. Append the dummy variables as columns to `df_1`
3. Import the NumPy library and use the function `np.log()` to compute the natural logarithm of exoplanet mass, radius, equilibrium temperature, and orbital period,. Assign the results to new columns in `df_1`

```
# Load the exoplanets data from the current directory

```

```
# Construct dummies for discovery facilities

# Compute the natural logarithm of mass, radius, temperature, and orbital period
```

**7.2** Use the empty code cell below to do the following:

1. Estimate the following multiple linear regression:

$$ln(M) = \beta_0 + \beta_1 ln(R) + \beta_4 ln(T) + \beta_5 ln(O) + \sum_{k=1}^{6} \gamma_q F_q + \varepsilon$$

   where $M$ is exoplanet mass, $R$ is raidus, $T$ is equilibrium temperature, and $F_q$ for $q = 1, 2, ..., 6$ are dummies for the discovery facilites. Put the coefficients in a new DataFrame and output it. Compute the RSE and the R^2 and print them.

2. Create a residual plot for model fit and output it

3. Take a moment to think about the following question: Does this model fit the data better than the linear regression models for exoplanet mass presented in the lecture? If so, what is the reason for this? The exploratory data analysis of the exoplanets data presented in lecture 5 can help you to answer this question

```
# Fit a multiple linear regression model and output the coefficients
```

```
# Compute goodness of fit metrics
```

```
# Create a residual plot and output it
```