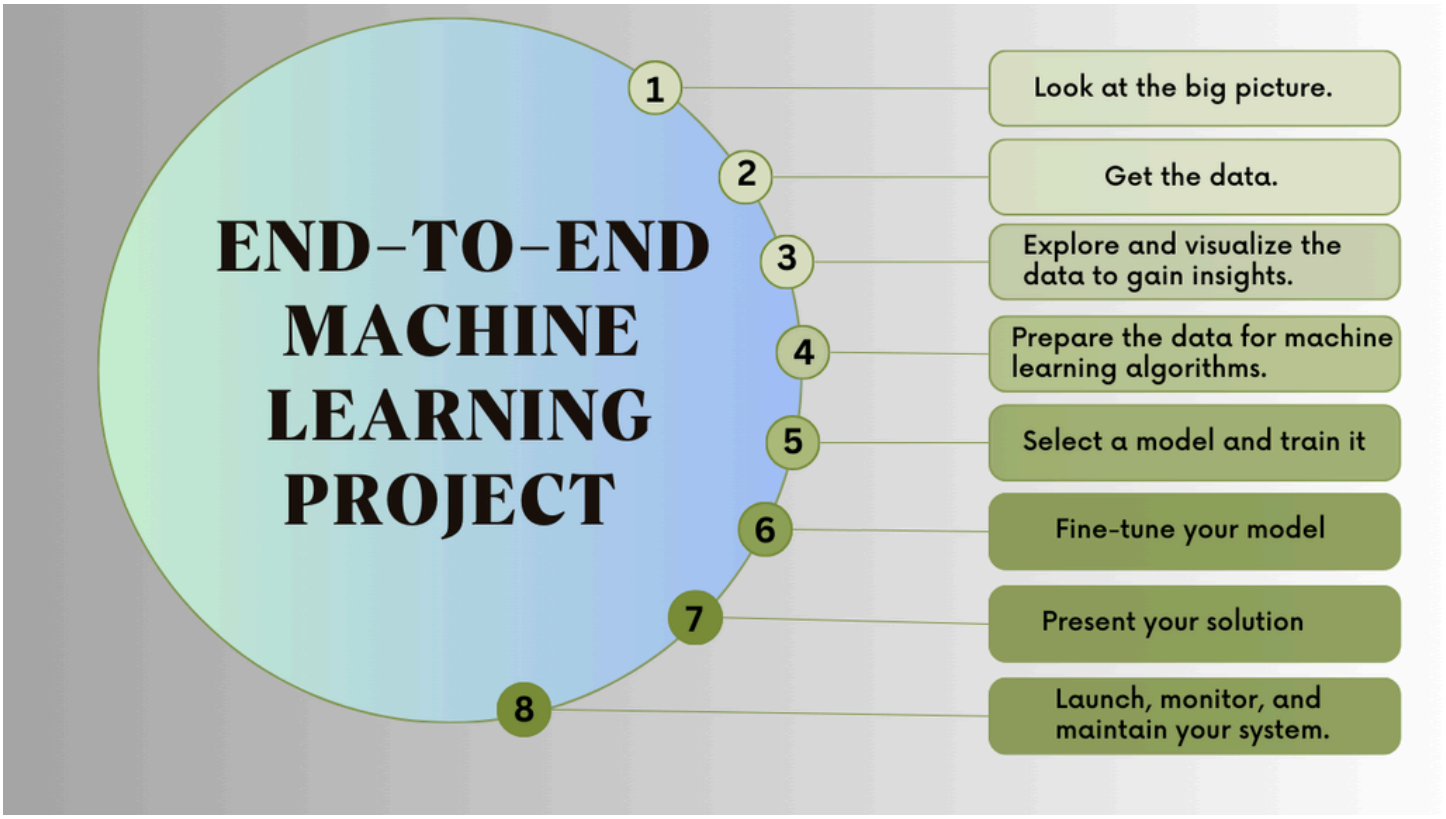
End-to-End Machine Learning Project

We will learn machine learning by doing an example project end to end. Suppose we are hired recently as a data Scientist at a real estate company. Our main goal is to learn main steps of machine learning not the real estate business.



a. We will use the California Housing prices data from the statlib repository

b. Suppose this is a recent data though its not

c. Let's make the dataset simple for teaching purpose. And added categorical attribute and remove a few features.

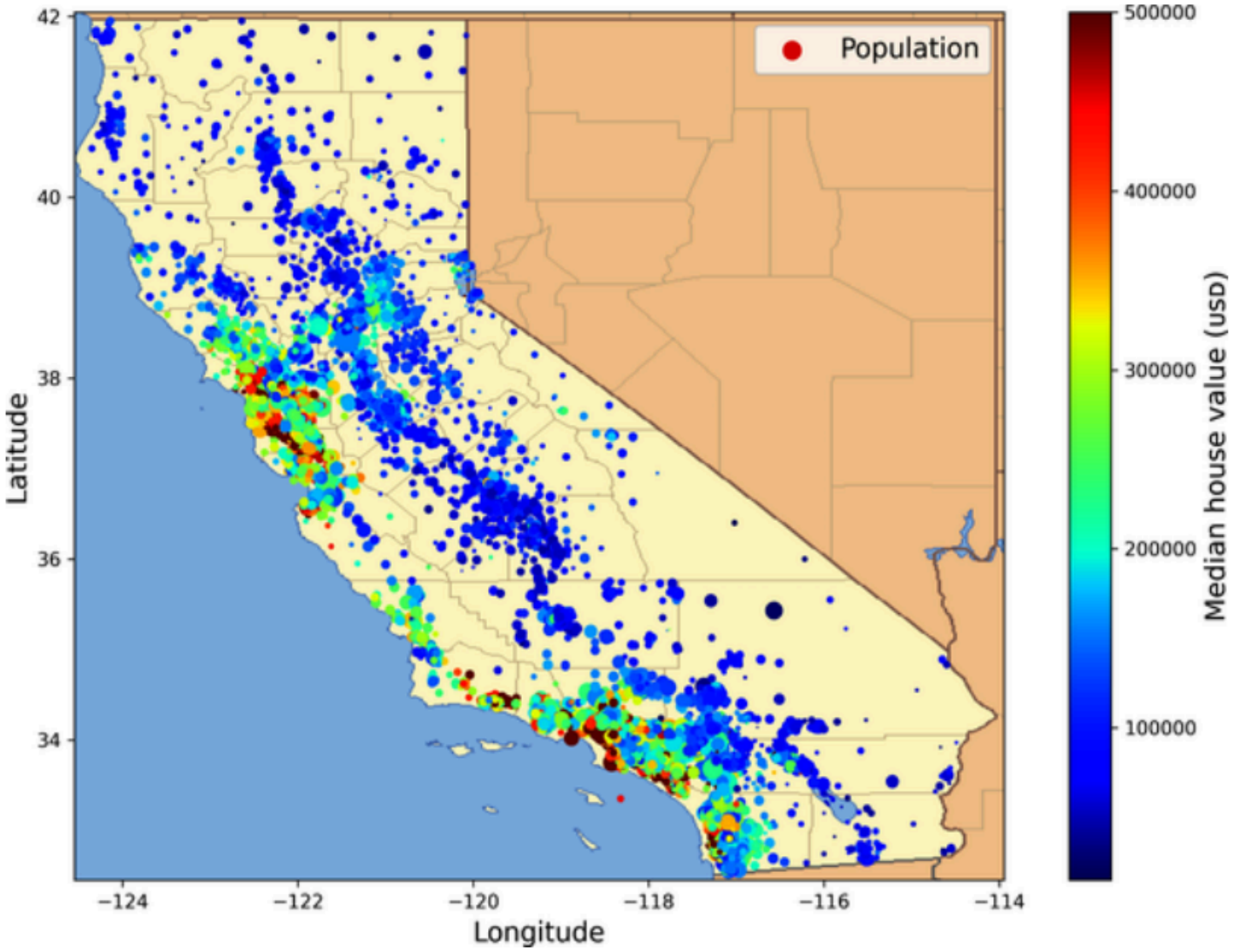


Figure 2-1. California housing prices

Look at the Big Picture

a. welcome to the company.

b. **First Task**: Build a model of housing prices in the sate by using California census data.

c. The metrics of the data such as

1. population
2. median income
3. median housing price for each block group in California.
4. Block Group: Smallest geographical unit(unit has a population of 600 to 3000)
5. Let's call them districts for short.

Expectations from the Model

The model should learn from the data and be able to predict the median housing price in any district.

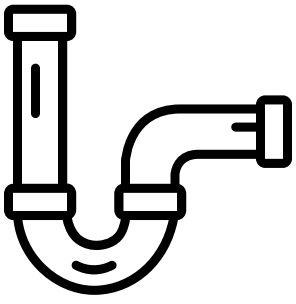
Since you are a well-organized data scientist, the first thing you should do is pull out your machine learning project checklist. You can start with the one in Appendix A; it should work reasonably

well for most machine learning projects, but make sure to adapt it to your needs. In this chapter we will go through many checklist items, but we will also skip a few, either because they are self-explanatory or because they will be discussed in later chapters.

Frame the Problem

1. First we have to ask what exactly the business objectives is.
2. Building model is probably not the end goal.
3. The goal is how does the company expect to use benefits from the model?
4. To know the objectives it's important to frame the problem which will help us to find out the right algorithm selection, will help in performance measure to evaluate the model and how much effort we will spend tweaking it.

Pipelines



Data Pipeline: A sequence of data processing components is called a data pipeline

Components typically run asynchronously.

Now we are ready to start designing our system.

Firstly,

We have to think what kinds' of training supervision the model will need?

1. **Supervised**
2. **Unsupervised**
3. **Semi-Supervised**
4. **Self supervised**
5. **Reinforcement learning task?**

Then,

We have to think is it

1. **Classification task or**
2. **Regression Task or some thing else?**

Then,

1. **should we use batch learning or**
2. **online learning technique ?**

Let’s find out answer

1. This is typical supervised learning task.
2. Since the model can be trained with **labeled examples**(each instance come's with the expected output i.e district median housing price)
3. It's typical regression task since the model will ask to predict a value
4. More specifically, this is **multiple regression** problem**,** Since system will use multiple features to make a prediction(the district populations, the median income etc.)
5. It's also **univariate regression** problem, since we are only trying to predict a single value for each district.
6. If we were trying to predict multiple value per districts, it would be a **multivariate regression** problem.
7. **Finally**, there is no continuous flow of data coming into the system, there is no particular need to adjust to changing data rapidly, the is small enough to fit the memory so plain **batch learning** should do just fine.

If the data were huge, you could either split your batch learning work across multiple servers (using the MapReduce technique) or use an online learning technique.

Select a performance measure

1. Now we have to select performance measure
2. A typical performance measure for regression problem is the **Root Mean Square Error (RMSE)**
3. It's gives an idea of how much error the system typically makes in it's predictions with a higher weight given to large error.

Equation 2-1. Root mean square error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

Notation

This equation introduces several very common machine learning notations that I will use throughout this book:

m is the number of instances in the dataset we are measuring the RMSE on. For example, if you are evaluating the RMSE on a validation set of **2,000** districts, then **m = 2,000.**

x is a vector of all the feature values (**excluding the label**) of the i instance in the dataset, and y is its label (the desired output value for that instance). For example, if the first district in the dataset is

located at longitude –118.29°, latitude 33.91°, and it has 1,416 inhabitants with a median income of $38,372, and the median house value is $156,400 (ignoring other features for now), then:

$$\mathbf{x}^{(1)} = \begin{pmatrix} -118.29 \\ 33.91 \\ 1{,}416 \\ 38{,}372 \end{pmatrix}$$

and:

$$y^{(1)} = 156{,}400$$

X is a matrix containing all the feature values (excluding labels) of all instances in the dataset. There is one row per instance, and the **i row is equal to the transpose of x , noted (x)** . For example, if the first district is as just described, then the matrix **X** looks like this:

$$\mathbf{X} = \begin{pmatrix} \left(\mathbf{x}^{(1)}\right)^{\mathsf{T}} \\ \left(\mathbf{x}^{(2)}\right)^{\mathsf{T}} \\ \vdots \\ \left(\mathbf{x}^{(1999)}\right)^{\mathsf{T}} \\ \left(\mathbf{x}^{(2000)}\right)^{\mathsf{T}} \end{pmatrix} = \begin{pmatrix} -118.29 & 33.91 & 1{,}416 & 38{,}372 \\ \vdots & \vdots & \vdots & \vdots \end{pmatrix}$$

h is your system's prediction function, also called a **hypothesis.** When your system is given an instance's feature vector **x** , it outputs a predicted value **$\hat{y}$ = h(x)** for that instance (**$\hat{y}$** is pronounced "**y-hat**").

For example, if your system predicts that the median housing price in the first district is **$158,400, then $\hat{y}$ = h(x) = 158,400**. The prediction error for this district is **$\hat{y} - y = 2,000$**. **RMSE(X,h)** is the cost function measured on the set of examples using your **hypothesis h**.

We use lowercase italic font for scalar values (such as **m or y**) and function names (such as **h**), **lowercase bold** font for vectors (such as **x**), and **uppercase** bold font for matrices (such as **X**).

Although the RMSE is generally the preferred performance measure for regression tasks, in some contexts you may prefer to use another function. For example, if there are **many outlier** districts. In that case, you may consider using the **mean absolute error (MAE, also called the average absolute deviation)**, shown in Equation 2-2:

Equation 2-2. Mean absolute error (MAE)

$$\mathrm{MAE}(\mathbf{X}, h) = \frac{1}{m}\sum_{i=1}^{m}\left|h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right|$$

Both the **RMSE** and the **MAE** are ways to measure the distance between two vectors: the vector of **predictions** and the vector of **target values**. Various distance measures, or norms, are possible: Computing the root of a sum of squares (RMSE) corresponds to the

Euclidean norm: this is the notion of distance we are all familiar with. It is also called the ℓ **norm**, noted $\|\cdot\|$ (**or just** $\|\cdot\|$). Computing the sum of absolutes (**MAE**) corresponds to the ℓ **norm**, **noted** $\|\cdot\|$. This is sometimes called the **Manhattan norm** because it measures the distance between two points in a city if you can only travel along orthogonal city blocks. More generally, the ℓ norm of a vector **v** containing **n** elements is defined as $\|\mathbf{v}\| = (|\mathbf{v}| + |\mathbf{v}| + \ldots + |\mathbf{v}|)$. ℓ gives the number of **nonzero elements** in the vector, and ℓ gives the **maximum absolute value** in the vector. The higher the norm index, the more it focuses on large values and neglects small ones. This is why the **RMSE is more sensitive to outliers** than the **MAE**. But when outliers are exponentially rare (like in a bell-shaped curve), the RMSE performs very well and is generally preferred.

Check the Assumptions

1. **List and Verify Assumptions**:
   - It's essential to identify and confirm any assumptions made so far to catch potential issues early.
2. **Check How Outputs Are Used**:
   - Understand how your model's output (predicted housing prices) will be used in downstream processes.
   - For example, if the downstream system converts prices into categories (like “cheap,” “medium,” or “expensive”), a classification approach may be more appropriate than regression.
3. **Avoid Misalignment**:

- Ensure that the problem is correctly framed to match the actual needs. This prevents wasting time on an unsuitable approach (e.g., using regression when classification is needed).

4. **Confirm Requirements with the Team**:
   - After consulting with the downstream team, verify that they need the actual prices, not categories.
   - In this case, they confirmed they need actual prices, so a regression approach is correct.
5. **Proceed with Confidence**:
   - With all assumptions checked and validated, you can confidently move forward with the coding phase.

Working with Real Data

1. **Importance of Real-World Data:**

- Experimenting with real-world data is more beneficial for learning machine learning than using artificial datasets.

2. **Popular Open Data Repositories:**

- **OpenML.org**

- **Kaggle.com**

- **PapersWithCode.com**

- **UC Irvine Machine Learning Repository**

- **Amazon’s AWS datasets**

- **TensorFlow datasets**

3. **Meta Portals Listing Open Data Repositories**:

- **DataPortals.org**

- **OpenDataMonitor.eu**

4. **Other Resources Listing Popular Open Data Repositories**:

- **Wikipedia’s list of machine learning datasets**

- **Quora.com**

- **The datasets subreddit**