Introduction to machine learning I

1. Marriage of machine learning and social science: Gebru et al. (2017)

1.1 Two social science questions

A social science question

> **how to estimate the demographic makeup of neighborhoods across the United States?**

and an even more interesting question

> **how does the demographic makeup of neighborhoods affect the presidential election?**

Social scientists might have been studying such problems for decades. And now another community also become interested in such problems and come with some new data, new idea and new methods. That is machine leanring community.

1.2. Answers from machine learning researchers

Answer to the above questions from some machine learning researchers in Stanford:

> **Cars**

Why?

- *Popularity*: cars are everywhere in every district of the United States
- *Diversity*: cars are diverse enough to reflect the different taste of people (saving/consuming, altitute to foreign countries, sensitivity to the income change, etc)

Next questions: **how to get data about cars?**

A naive answer would be a census of the cars of people in every district of USA which will cost huge money and resources. (every year USA spends over 1 billion dollars on census).

Another answer

> use the **auto-detection** and **classification** techniques from machine learning and computer vision

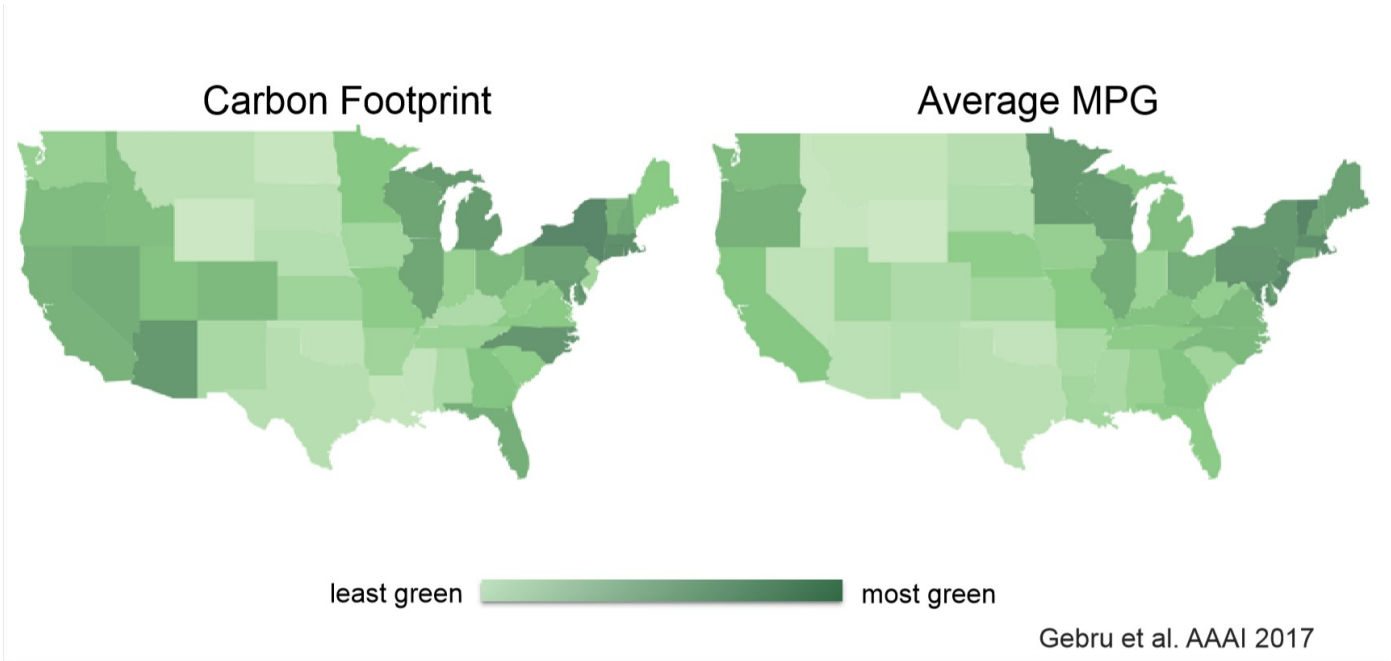
more specifically,

> Using **deep learning-based computer vision techniques** to estimate socioeconomic characteristics of regions spanning 200 US cities by using 50 million images of street scenes gathered with Google Street View cars

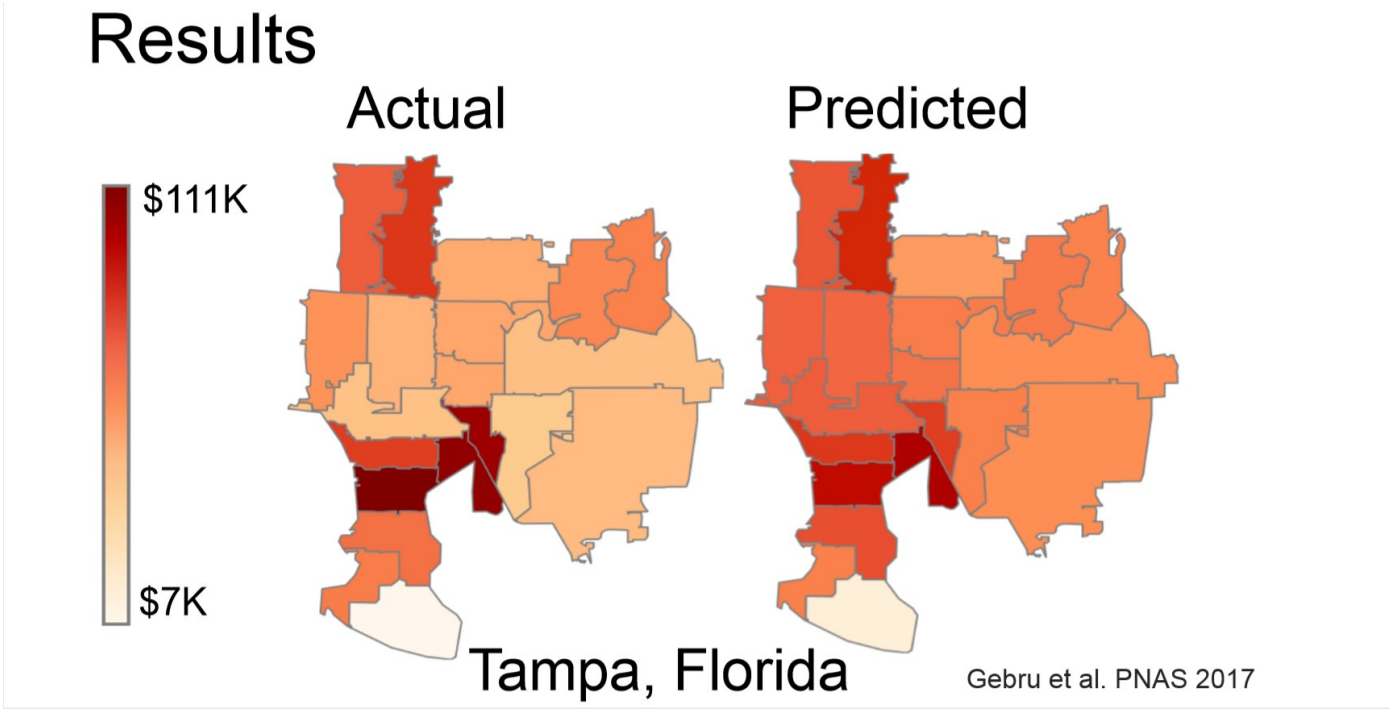
1.3. The effect of machine learning techniques

Gebru et al. (2017) create a data set of cars in the 50 million Google view images using deep learning techniques. It is free to download here [Visual Census: Fine-Grained Car Dataset (https://ai.stanford.edu/~tgebru/car_data.html)](https://ai.stanford.edu/~tgebru/car_data.html)

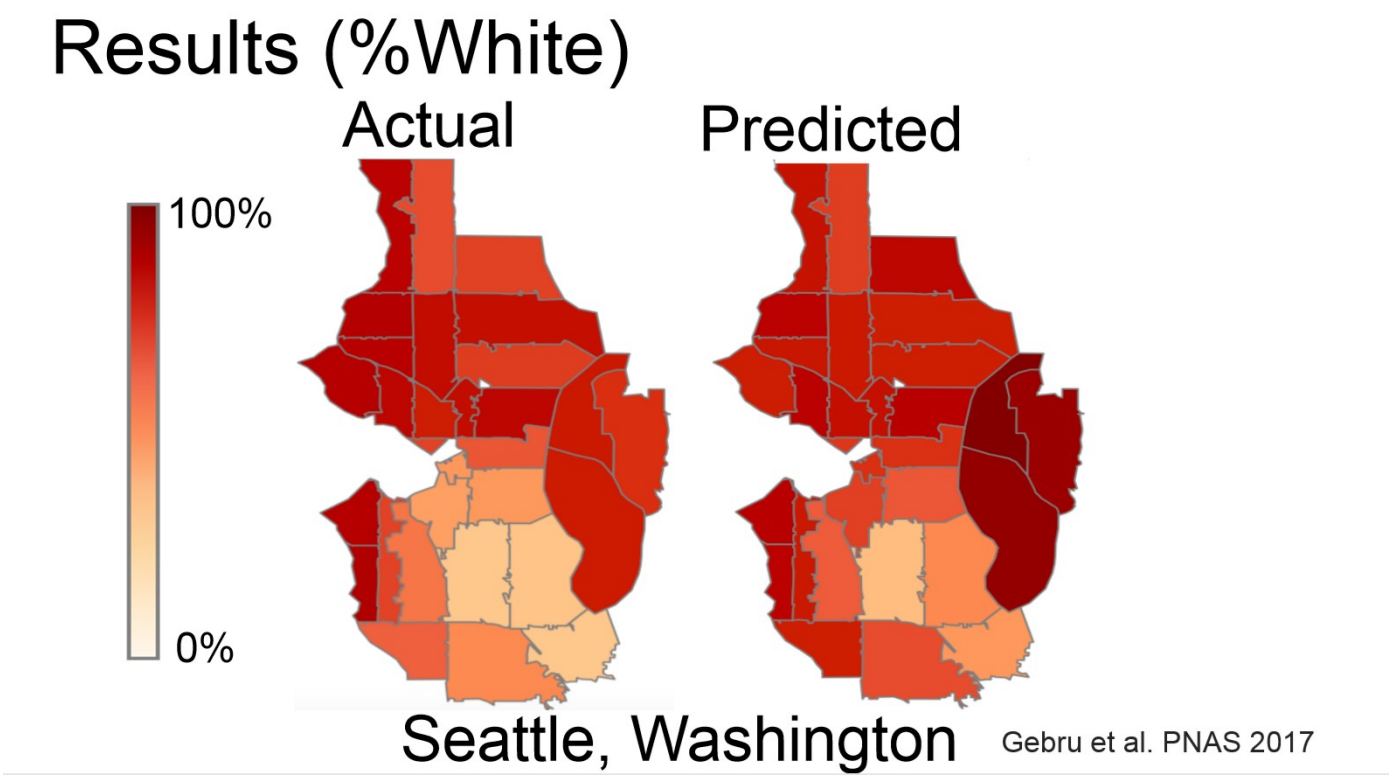
Q1: How green is each state?



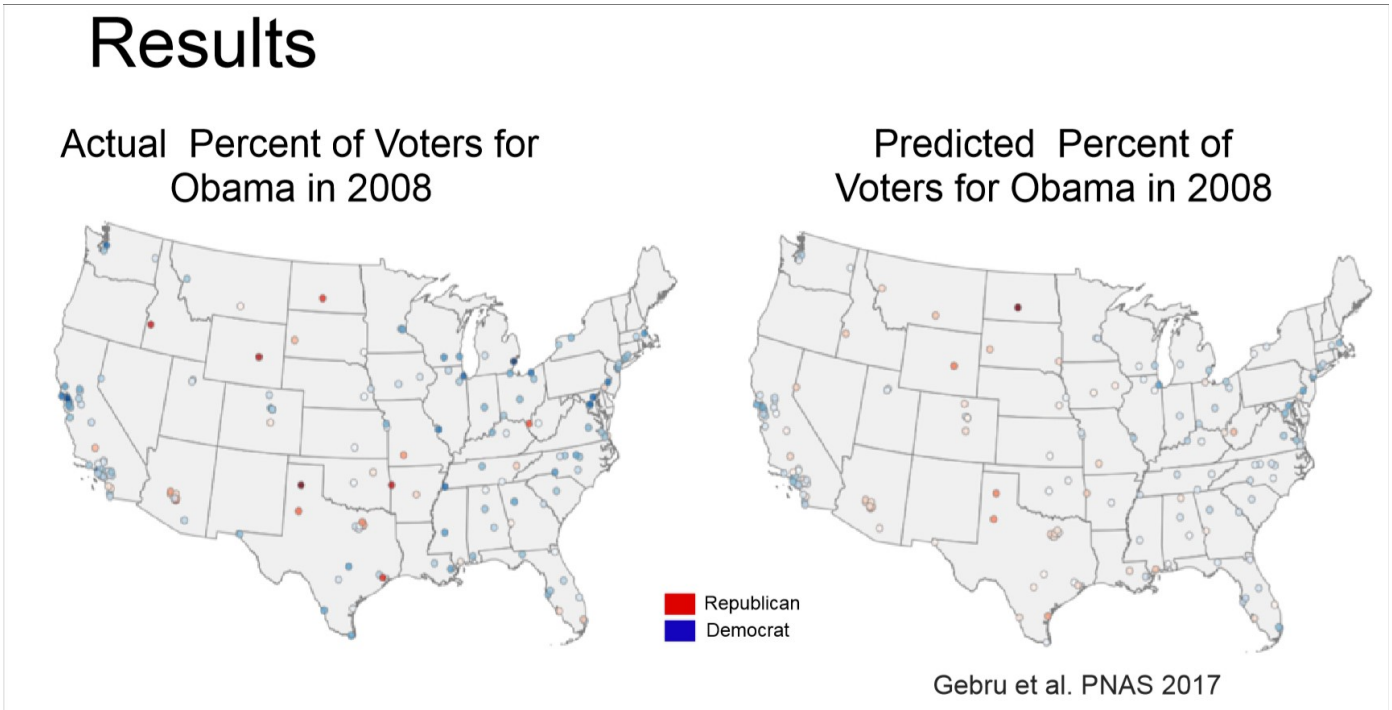
Q2: Can we predict the income?



Q3: Can we predict the race distribution?



Q4: Can we predict the voting pattern?



Now we will study the techniques behind such great success: **supervised learning**

2. Supervised learning

> Supervised learning is the machine learning task of inferring a function from labeled training data.

2.1. Mathematical definition

Mathematically, it means there is an unobservable function $f : X \to Y$ where X is the input, Y is the label. Given the data $(X_1, Y_1), \ldots, (X_n, Y_n)$, we want to use an algorithm to output an estimate of the function $\hat{f}$.

2.2. Basic ideas

Training, testing data

Usually, we will divide the data set into 3 parts:

- *training data* : learn the parameters of the model (sometimes, also hyperparameters)
- *testing data* : estimate the performance of the model

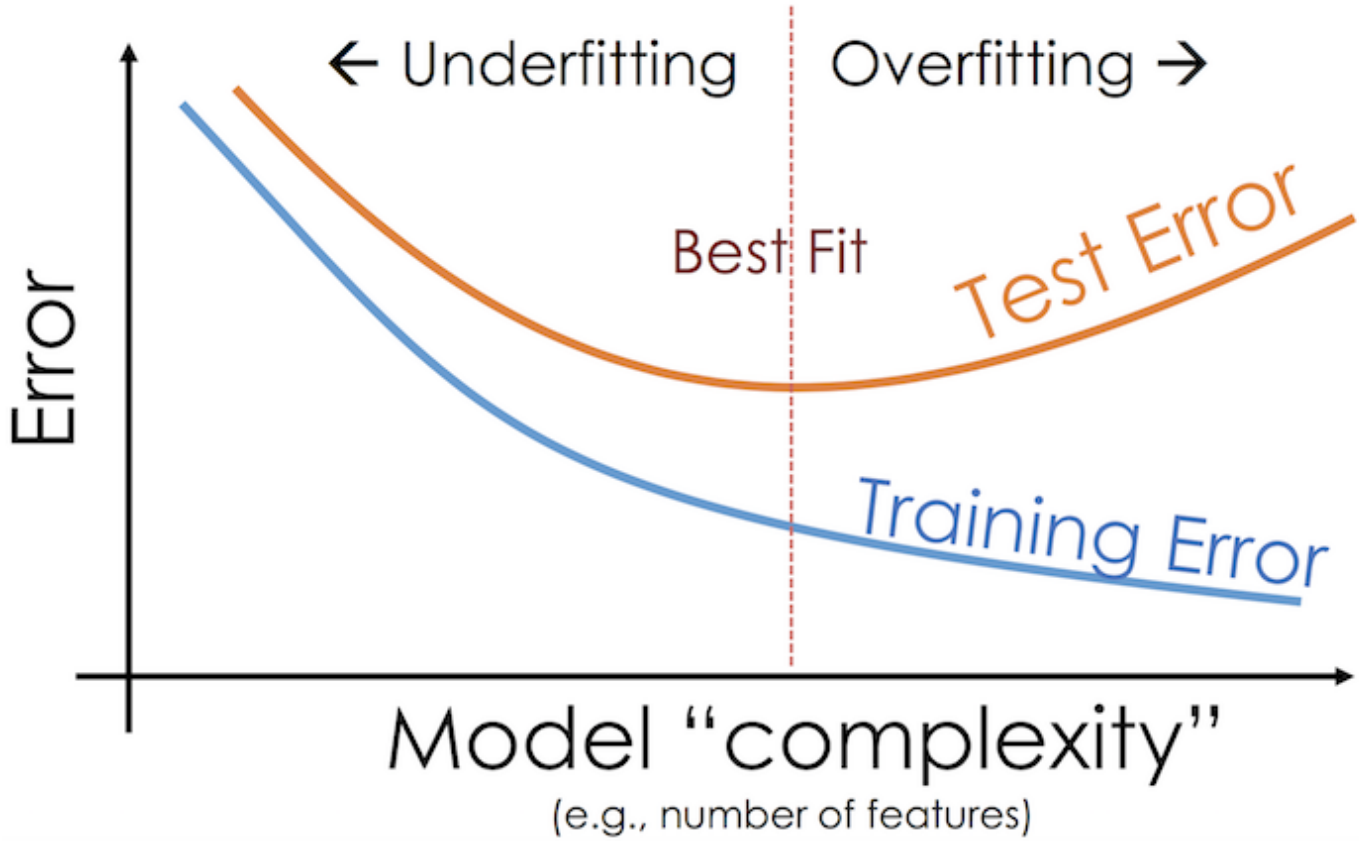
Prediction accuracy

Supervised learning can be divided into two catergories based on the output of this function:

- regression problem: $Y \in \mathbb{R}$
- classfication problem: $Y \in \{0, 1, \ldots, n\}$

In machine learning, the prediction accuaracy is the mean squared error between the predicted values and the real values in the regression problem and the classification correctness rate for the classification problem.

Generalisation error and overfitting



There have been many methods nowadays:

- linear regression
- nearest neighbor
- support vector machine
- random forest
- gradient boosting machine
- neural network (aka deep learning)

all these methods can be found in `sklearn` package which is the one of the most popular machine learning packages in Python.

Anaconda has installed `sklearn` module.

3. Typical machine learning algorithms in `sklearn`

3.1 Data set

we use the `Guerry` data set from `statsmodels` to study a regression problem: can we predict the sales of the lottery?

In [3]:

```
import statsmodels.api as sm

# Load data
dat = sm.datasets.get_rdataset("Guerry", "HistData").data
# remove all the non-numerical columns
dat.drop(["dept", "Region", "Department", "MainCity"], axis=1, inplace=True)
# list of the variables
dat.head(5)
```

Out[3]:

	Crime_pers	Crime_prop	Literacy	Donations	Infants	Suicides	Wealth	Commerce	Clergy
0	28870	15890	37	5098	33120	35039	73	58	11
1	26226	5521	51	8901	14572	12831	22	10	82
2	26747	7925	13	10973	17044	114121	61	66	68
3	12935	7289	46	2733	23018	14238	76	49	5
4	17488	8174	69	6962	23076	16171	83	65	10

The detailed information about the data set can be found here (https://rdata.pmagunia.com/dataset/r-dataset-package-histdata-guerry)

In [4]:

```
dat.describe()
```

Out[4]:

	Crime_pers	Crime_prop	Literacy	Donations	Infants	Suicides	W
count	86.000000	86.000000	86.000000	86.000000	86.000000	86.000000	86.0(
mean	19754.406977	7843.058140	39.255814	7075.546512	19049.906977	36522.604651	43.5(
std	7504.703073	3051.352839	17.364051	5834.595216	8820.233546	31312.532649	24.9(
min	2199.000000	1368.000000	12.000000	1246.000000	2660.000000	3460.000000	1.0(
25%	14156.250000	5933.000000	25.000000	3446.750000	14299.750000	15463.000000	22.2(
50%	18748.500000	7595.000000	38.000000	5020.000000	17141.500000	26743.500000	43.5(
75%	25937.500000	9182.250000	51.750000	9446.750000	22682.250000	44057.500000	64.7(
max	37014.000000	20235.000000	74.000000	37015.000000	62486.000000	163241.000000	86.0(

In [5]:

```
Y = dat['Lottery']
dat.drop('Lottery', axis=1, inplace=True)
```

We randomly divide the data set into `train_X`, `train_Y`, `test_X` and `test_Y` where the size of training, validation and test $\sim 4:1$

In [17]:

```
import numpy as np

train_X = []
train_Y = []
test_X = []
test_Y = []

for idx, row in dat.iterrows():
    r = np.random.randint(5)
    if  r < 4:
        train_X.append(row.values)
        train_Y.append(Y[idx])
    else:
        test_X.append(row.values)
        test_Y.append(Y[idx])

print("There are " + str(len(train_Y)) + " training data")
print("There are " + str(len(test_Y)) + " test data")
```

```
There are 70 training data
There are 16 test data
```

3.2. Linear model

In [18]:

```
# OLS
from sklearn import linear_model
from sklearn.metrics import mean_squared_error, r2_score
ols = linear_model.LinearRegression()
ols.fit(train_X, train_Y)
ols_pred_Y = ols.predict(test_X)

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(test_Y, ols_pred_Y))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(test_Y, ols_pred_Y))
```

```
Mean squared error: 428.82
Variance score: 0.42
```

In [19]:

```
# Lasso
from sklearn.linear_model import LassoCV
from sklearn.metrics import mean_squared_error, r2_score
lasso = LassoCV(cv=5, random_state=0)
lasso.fit(train_X, train_Y)
lasso_pred_Y = lasso.predict(test_X)

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(test_Y, lasso_pred_Y))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(test_Y, lasso_pred_Y))
```

```
Mean squared error: 524.26
Variance score: 0.29
```

3.3. k-Nearest Neighbor

In [22]:

```
from sklearn import neighbors
n_neighbors = 10
knn = neighbors.KNeighborsRegressor(n_neighbors, weights='uniform')
knn.fit(train_X, train_Y)
knn_pred_Y = knn.predict(test_X)

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(test_Y, knn_pred_Y))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(test_Y, knn_pred_Y))
```

```
Mean squared error: 587.54
Variance score: 0.20
```

3.4. Support vector regression

In [23]:

```
from sklearn import svm
clf = svm.SVR()
clf.fit(train_X, train_Y)
clf_pred_Y = clf.predict(test_X)

# The mean squared error
print("Mean squared error: %.2f"
      % mean_squared_error(test_Y, clf_pred_Y))
# Explained variance score: 1 is perfect prediction
print('Variance score: %.2f' % r2_score(test_Y, clf_pred_Y))
```

```
Mean squared error: 771.00
Variance score: -0.05

C:\Users\jason\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWa
rning: The default value of gamma will change from 'auto' to 'scale' in vers
ion 0.22 to account better for unscaled features. Set gamma explicitly to 'a
uto' or 'scale' to avoid this warning.
  "avoid this warning.", FutureWarning)
```