



# Data Science With Python

---

*Mosky*

# Data Science

---

- = Extract knowledge or insights from data.
- Data Science ⊃
  - Visualization
  - Statistics
  - Machine Learning
  - Big Data
  - Etc.
- ≈ Data Mining

# Statistics vs. Machine Learning

---

- Statistics constructs more solid **inferences**.
- Machine learning constructs more interesting **predictions**.
  - Machine Learning ⊇ Deep Learning
- The models may be the same, but the focuses are different.
- Good **predictions** usually needs good **inferences** on dataset.

# Science, Analysis, Scientist, and Engineering

---

- Data Engineering / Data Engineer
  - Prepare the data infra to enable others to work with.
- Data Analysis / Data Analyst
  - Analyze to help the company's decisions.
- Data Scientist
  - Create software to optimize the company's operations.



# Mosky

---

- Backend Lead at Pinkoi.
- Has spoken at: PyCons in TW, JP, SG, HK, KR, MY, COSCUPs, and TEDx, etc.
- Countless hours on teaching Python.
- Own Python packages: ZIPCodeTW, etc.
- <http://mosky.tw/>

# Outline

---

1. **Exploratory** (EDA, Exploratory Data analysis)
  - Correlation Analysis, PCA, FA, etc.
2. **Inference** (Statistical Inference)
  - Hypothesis Testing, OLS, Logit, etc.
3. **Preprocessing**
  - By pandas, scikit-learn, etc.
4. **Prediction** (Machine Learning Prediction)
  - SVM, Trees, KNN, K-Means, etc.
5. **Models of Models**
  - Cross Validation & Pipeline, Model Development, etc.

# PDF & Notebooks

---

- The PDF and notebooks are available here:
  - <https://github.com/moskytw/data-science-with-python>
- A good notebook reader:
  - <https://nbviewer.jupyter.org/>
- Or run it on your own computer:
  - Prepare Python and Pipenv.
  - `$ pipenv sync`

# Datasets

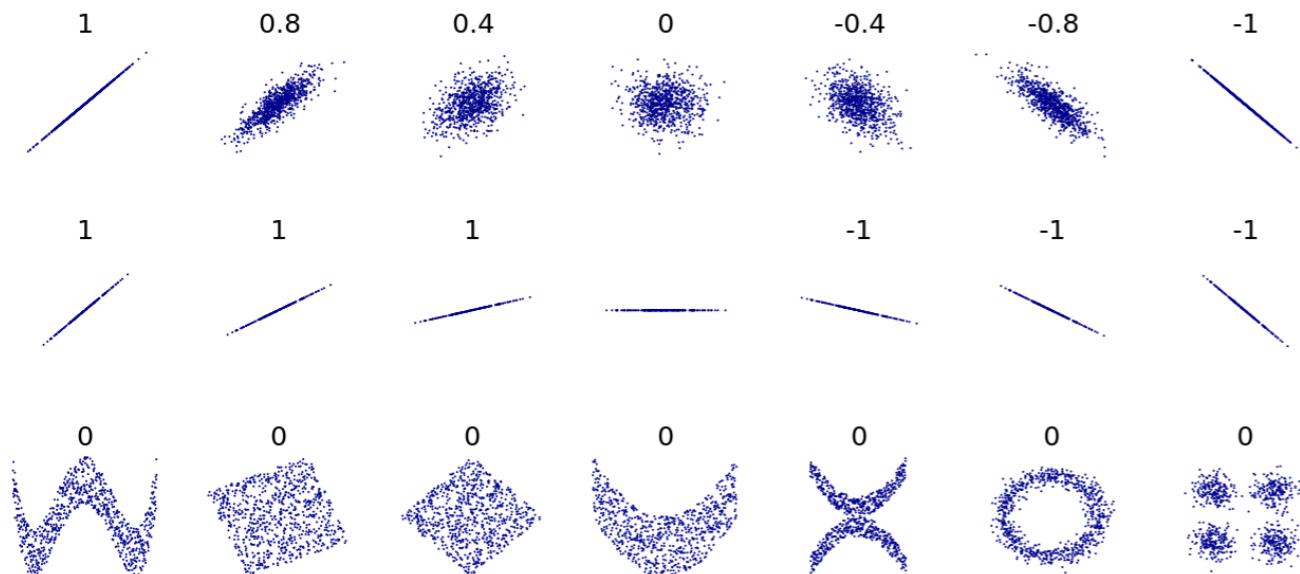
---

- The handouts are based on:
  - American National Election Survey 1996 ( $944 \times 10$ )
- You may play with:
  - Extramarital Affairs Dataset (1978;  $6366 \times 9$ )
  - Star98 Educational Dataset (1998;  $303 \times 13$ )
- Handout: datasets.ipynb
- The context matters:
  - 1970s – Wikipedia, 1990s – Wikipedia.
  - 1996 United States presidential election – Wikipedia.

# Exploration

# Correlation Analysis

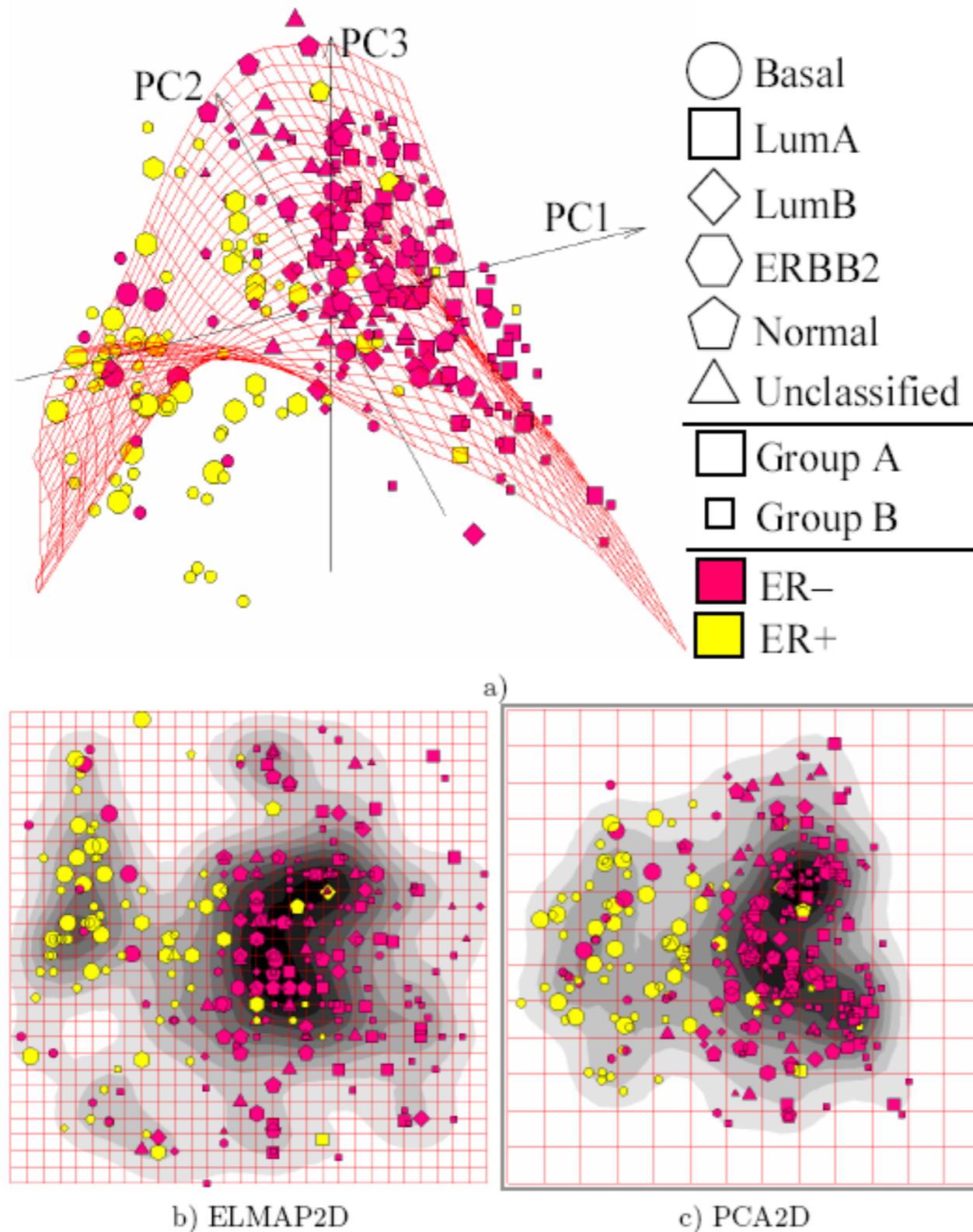
---



- Measures the bivariate linear “tightness”.
- ← Pearson's Correlation Coefficient ( $r$ )
- All pairs → correlation matrix.
- Handout:  
correlation\_analysis.ipynb

# PCA & FA

- .....
- Maps into a lower-dim space.
- ↙ Principal Component Analysis (PCA)
  - Visualize quickly.
- Factor Analysis (FA)
  - Assume lower-number unobserved variables (factors) exist.
- Handouts:
  - pca.ipynb, pca\_3d.ipynb, ipywidgets.ipynb, fa.ipynb



## See Also

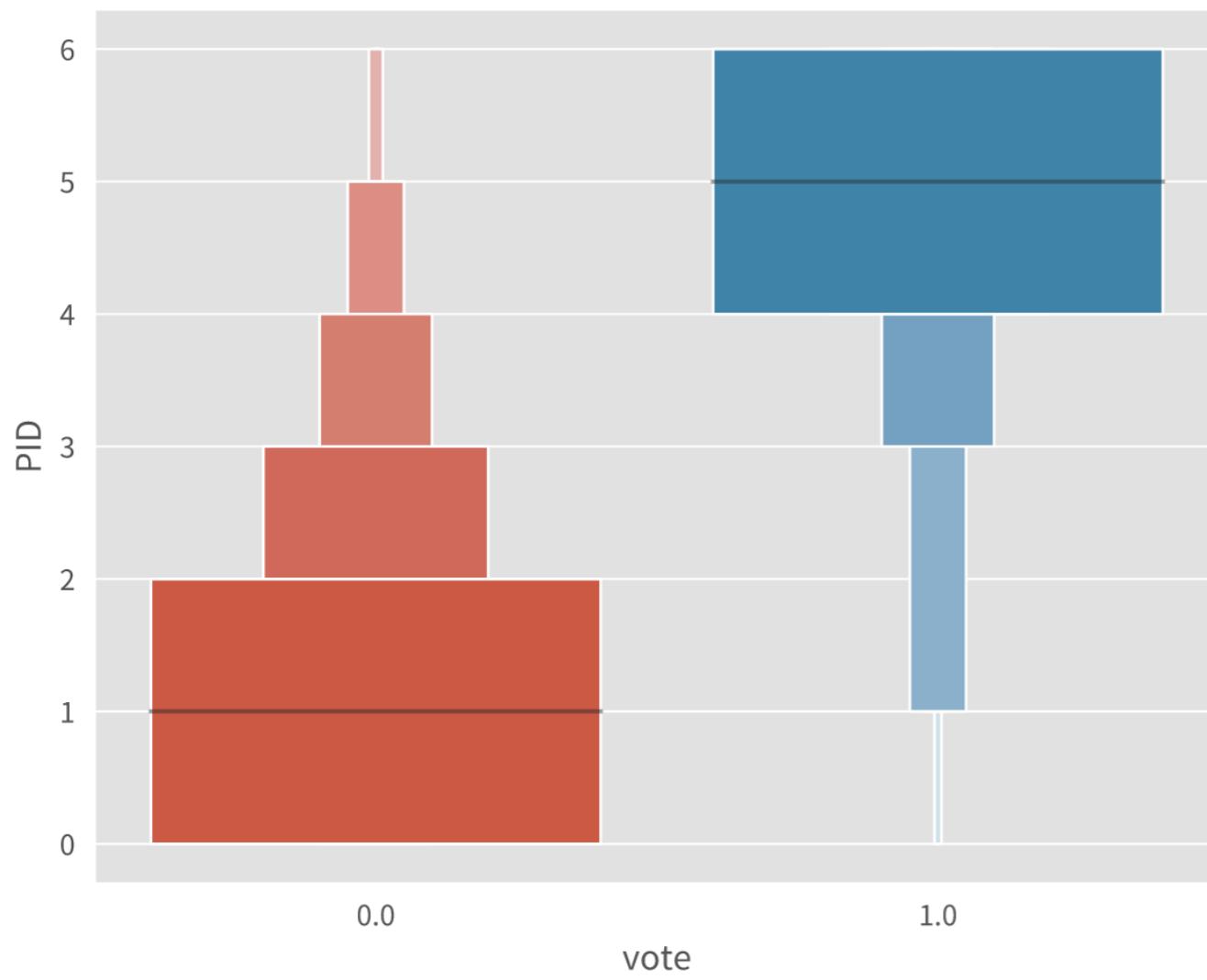
---

- [seaborn](#)
  - For drawing attractive and informative statistical graphics.
- [Plotly](#)
  - Makes interactive graphs.
- [pandas.DataFrame.corr](#)
  - Also has Kendall's  $\tau$  (tau) and Spearman's  $\rho$  (rho).
- [Isomap – scikit-learn](#)
  - Seeks a lower-dimensional embedding which maintains geodesic distances between all points.
- [Dimensionality reduction – scikit-learn](#)

# Inference

# Hypothesis Testing

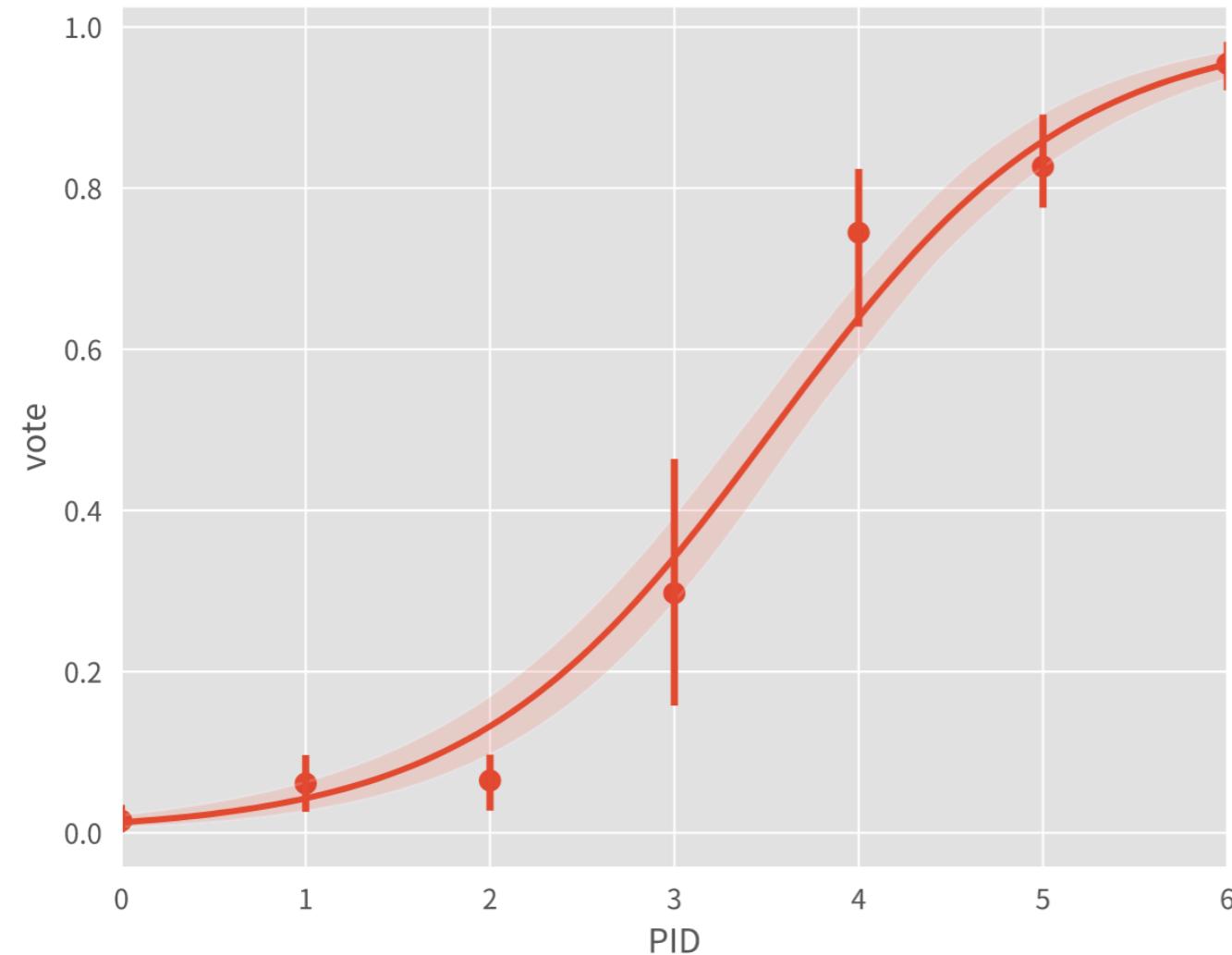
---



- Given a hypothesis, calculate the probability to observe the data.
- The hypothesis may be:
  - “the means are the same”
  - “the medians are the same”
  - “the prop. are the same, e.g., conversion rates”, etc.
- Like testing the performances of the model A and the B.
- Handout:  
`hypothesis_testing.ipynb`

# OLS & Logit

---



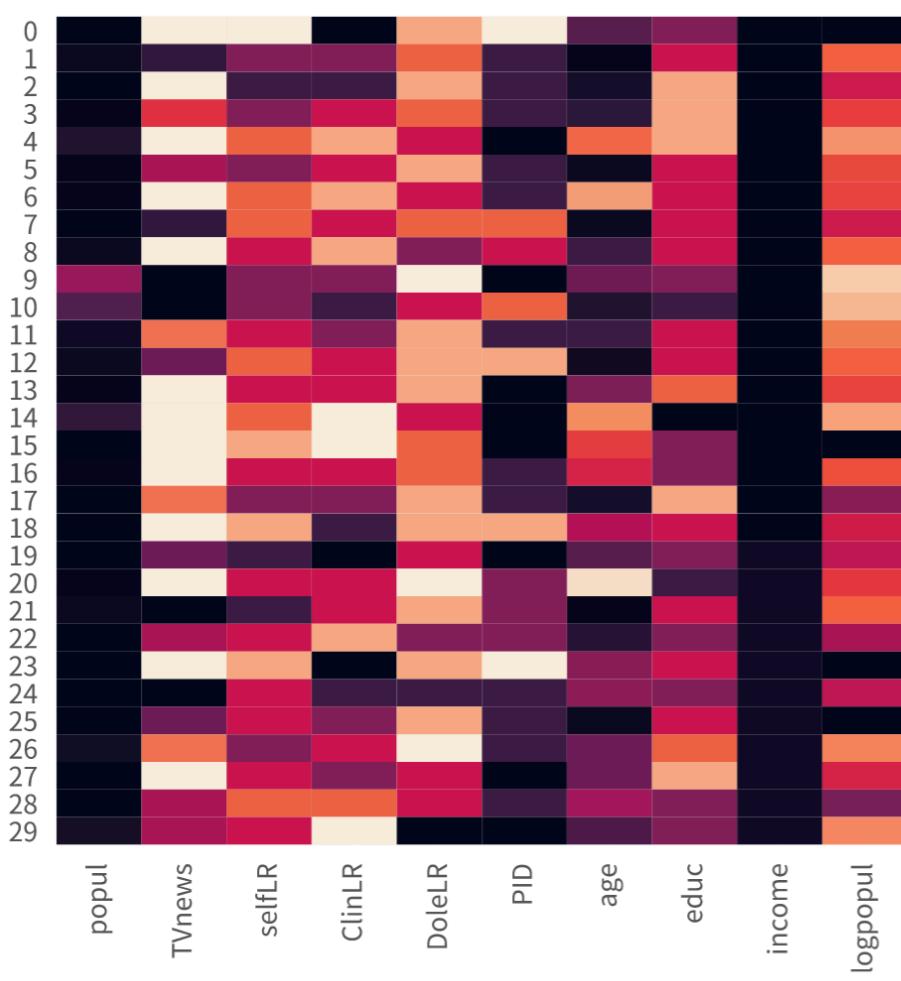
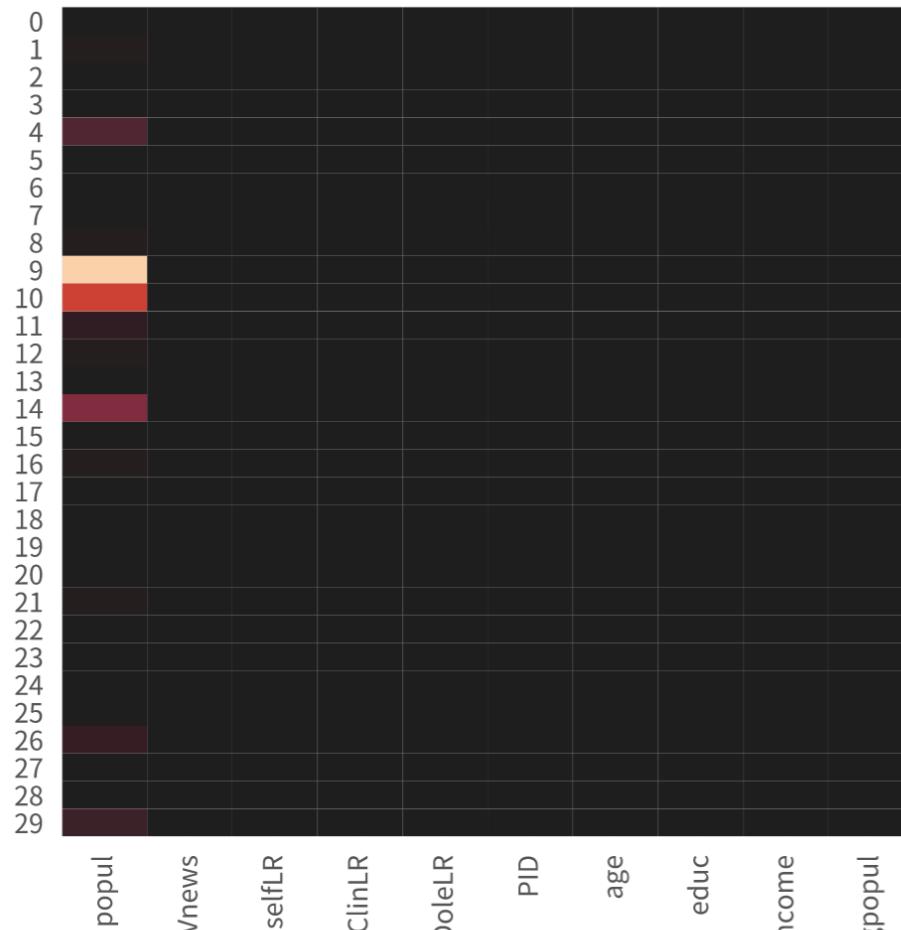
- Measures the “steepness”.
- With various assumptions:
  - Linear: OLS
  - $y \in \{0, 1\}$ : Logit
  - $y \in \{0, 1, \dots\}$ : Poisson, etc.
- ← Logit Regression
- Like understanding the dataset primarily, or even find the insights directly.
- Handouts:  
ols.ipynb, logit.ipynb

## See Also

---

- [Statistical functions – SciPy](#)
  - Includes most of the hypothesis testing functions.
- [User Guide – statsmodels](#)
  - Includes much more models for statistical inference.
- [Hypothesis Testing With Python](#)
  - Answers like “how much sample is enough?”
- [Statistical Regression With Python](#)
  - Answers like “how to read a regression summary?”

# Preprocessing



# Preprocessing

---

- Make the models understand the data by various methods.
- ← MixMinScaler
- Handouts:  
pandas\_preprocessing.ipynb,  
sqlite.ipynb,  
sklearn\_preprocessing.ipynb,

## See Also

---

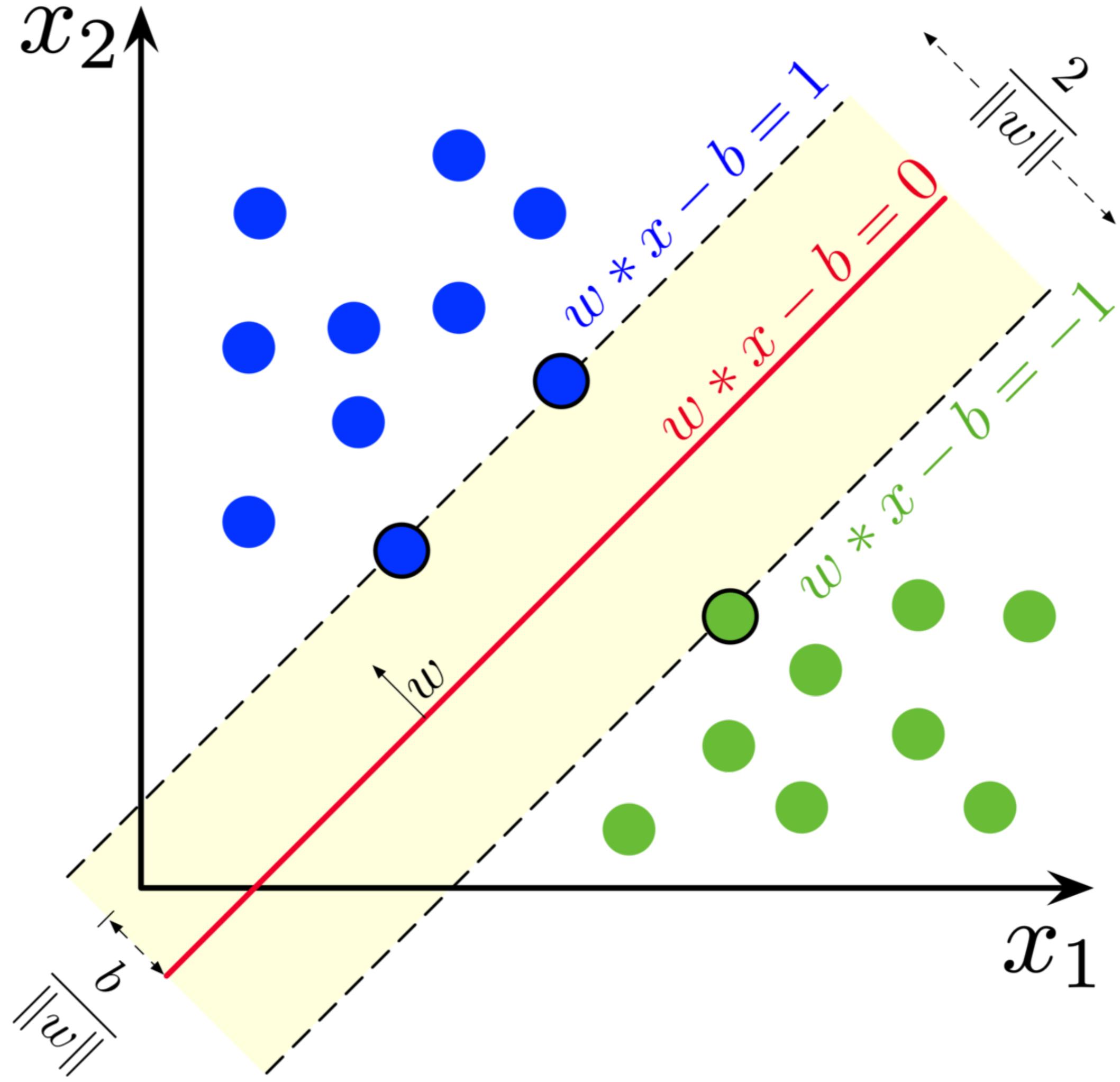
- [Text feature extraction & Image feature extraction](#) – scikit-learn
- [Column Transformer with Mixed Types](#) – scikit-learn
- [patsy](#): describe models by formulas, e.g.,  $y \sim age + C(gender)$ .
- [imbalanced-learn](#): balance the classes more carefully.
  - The `class_weight='balanced'` in scikit-learn may be also useful.
  - Since we may not want the model to be partial to the majority.
- Rather than pandas:
  - [dfply](#): Some consider cleaner.
  - [datatable](#): Faster.
  - [Spark](#): More scalable.
  - [Database-like ops benchmark](#) – H2O.ai
- [Feature Engineering](#): create features by domain knowledge.

# Prediction

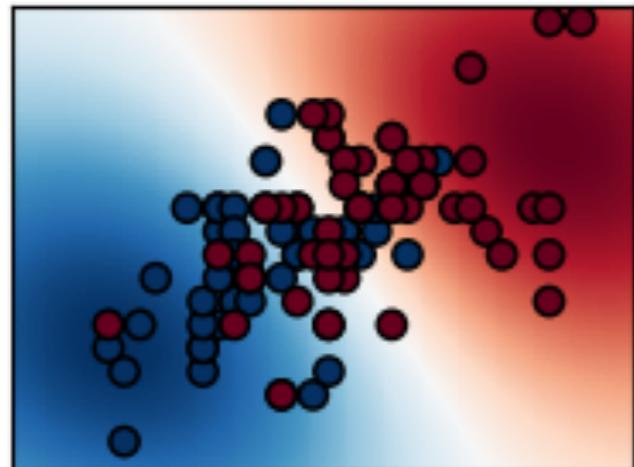
# Prediction

---

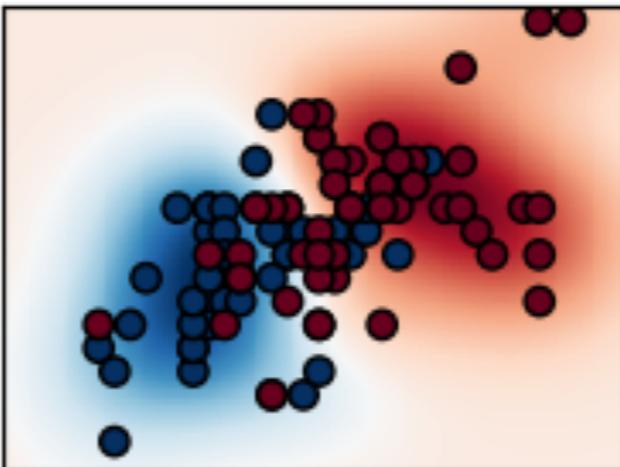
*Support-Vector Machines (SVM)*



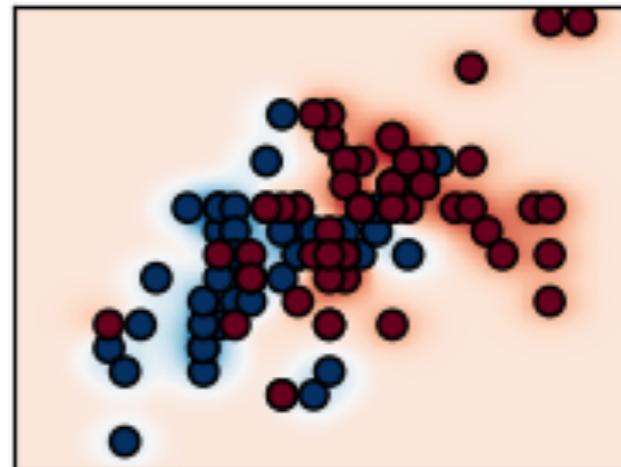
$\text{gamma}=10^{-1}, C=10^{-2}$



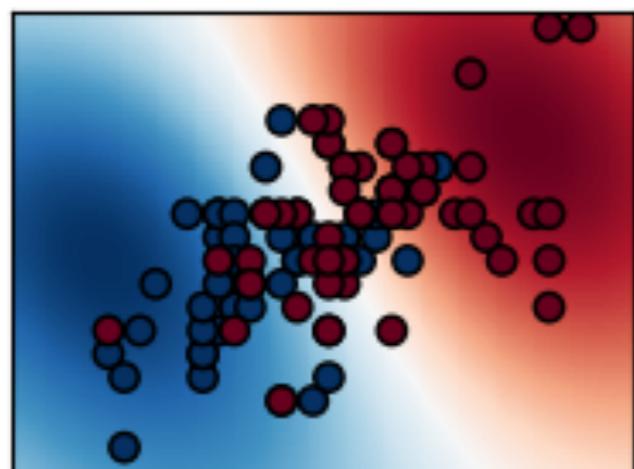
$\text{gamma}=10^0, C=10^{-2}$



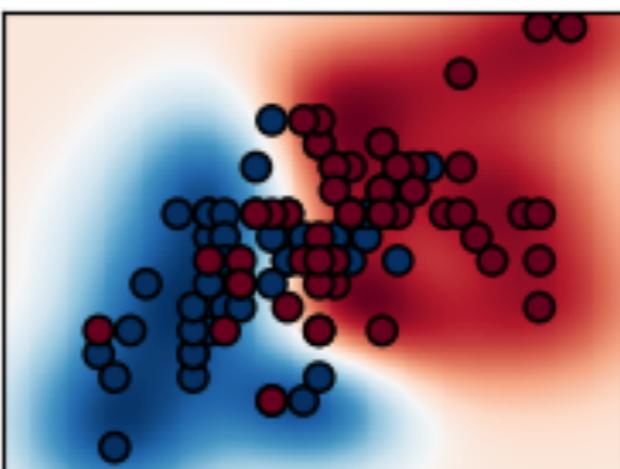
$\text{gamma}=10^1, C=10^{-2}$



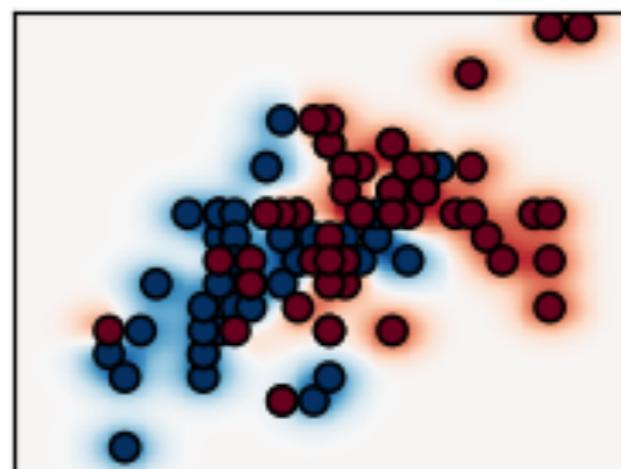
$\text{gamma}=10^{-1}, C=10^0$



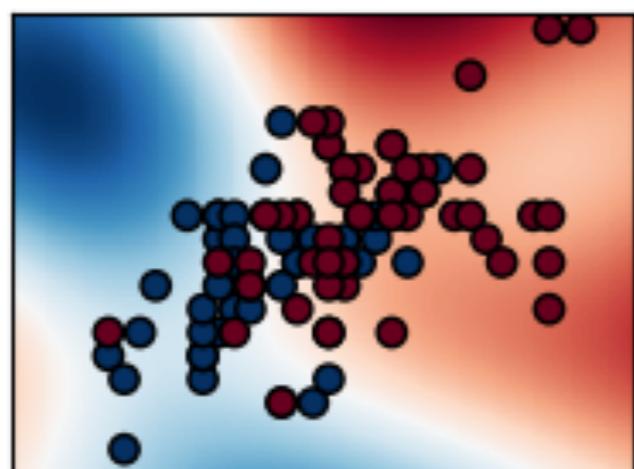
$\text{gamma}=10^0, C=10^0$



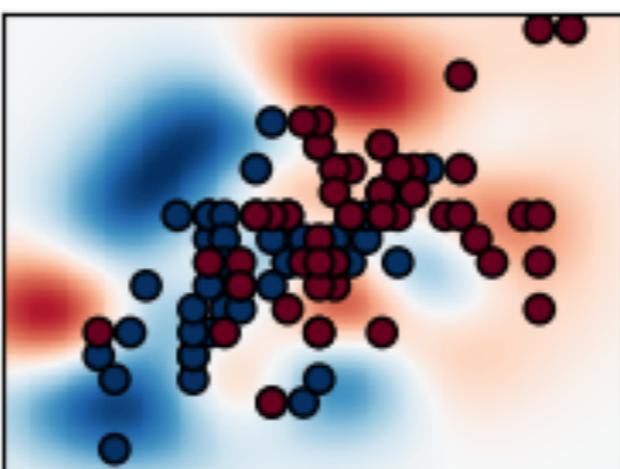
$\text{gamma}=10^1, C=10^0$



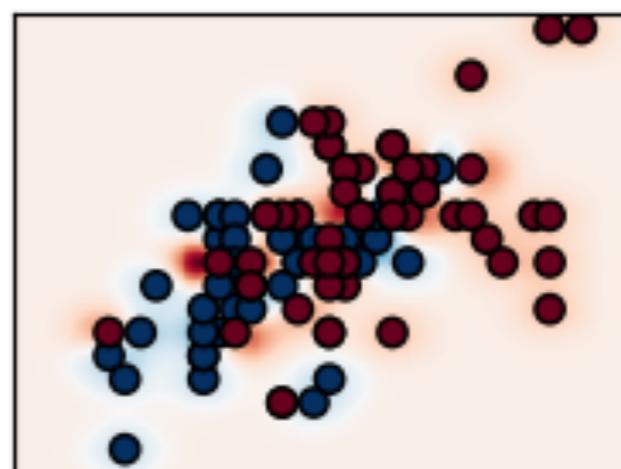
$\text{gamma}=10^{-1}, C=10^2$



$\text{gamma}=10^0, C=10^2$



$\text{gamma}=10^1, C=10^2$

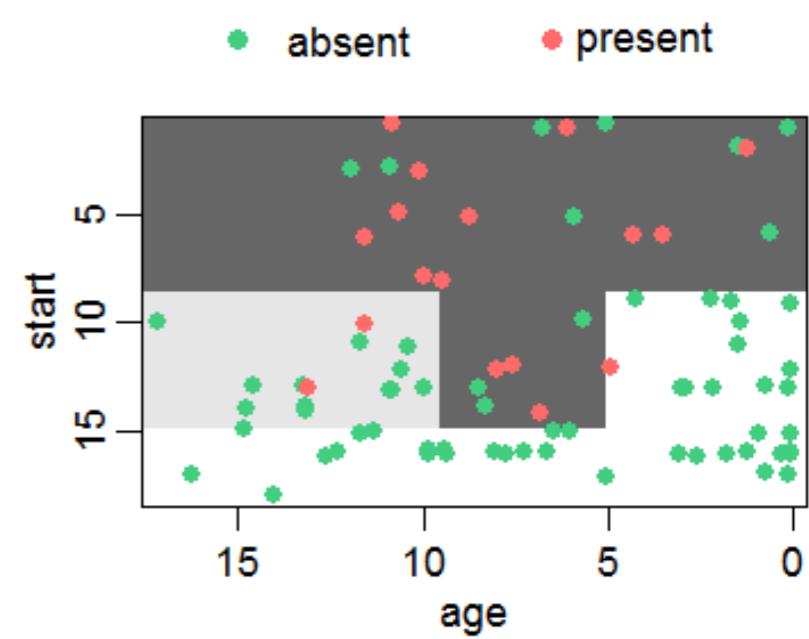
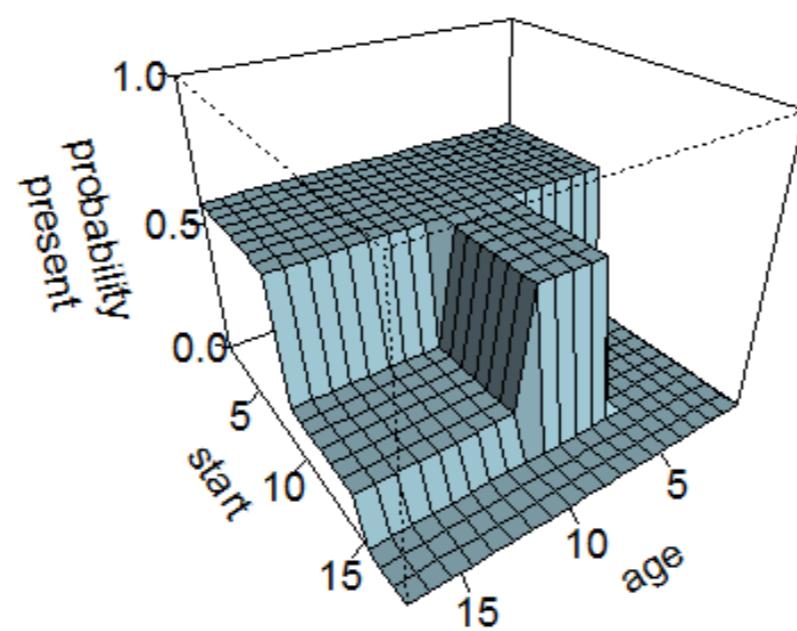
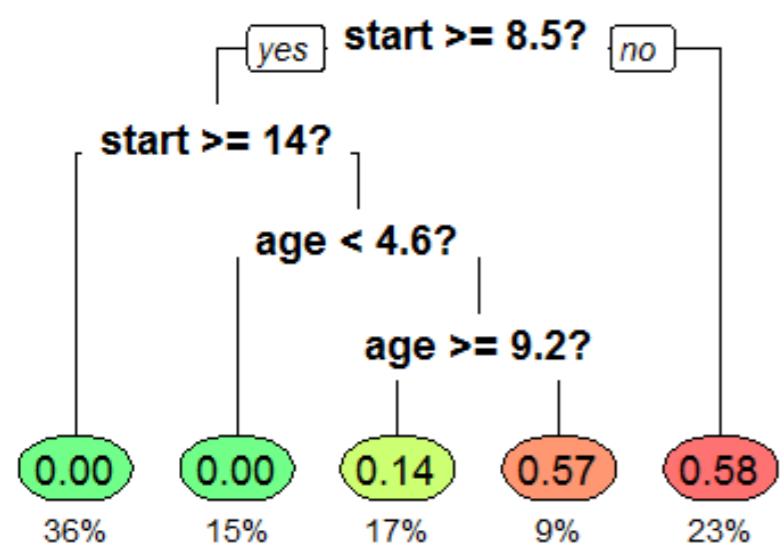


*SVM With Radial Basis Function (RBF) Kernel*

# Prediction

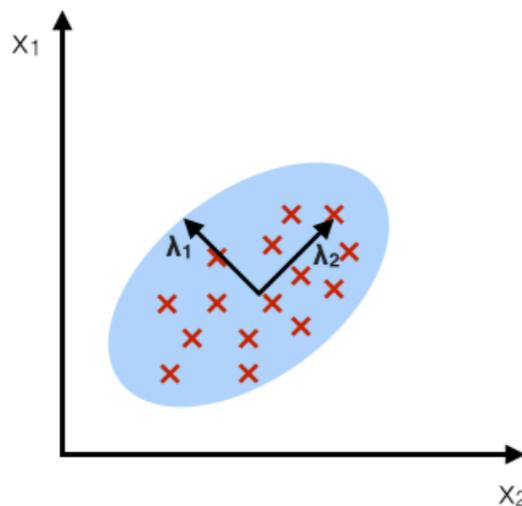
---

*Decision Tree*



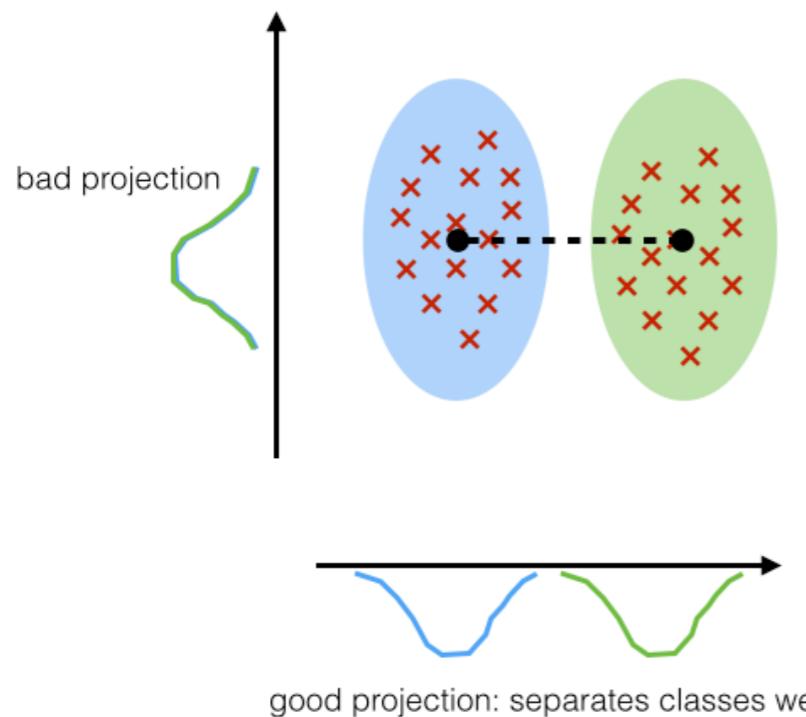
## PCA:

component axes that maximize the variance



## LDA:

maximizing the component axes for class-separation



## Prediction

.....

- Predict the category or continuous value.
- By various models:
  - ↑ SVM
  - ↑ Tree
  - ← Linear Discriminant Analysis (LDA)
- KNN & K-Means
- Handouts: svm.ipynb, trees.ipynb, logistic\_and\_lda.ipynb, knn.ipynb, kmeans.ipynb

## See Also

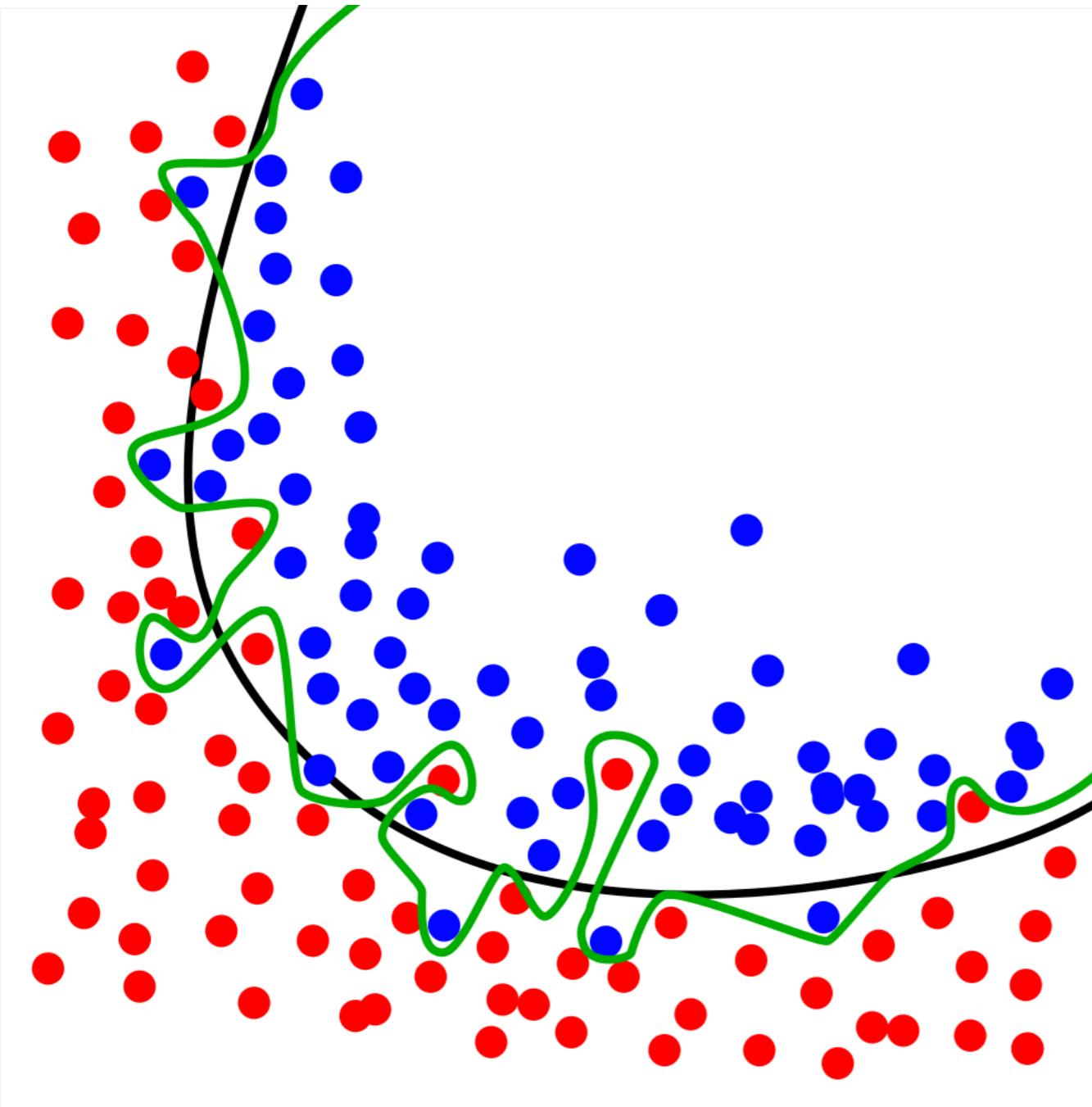
---

- [LightGBM](#): the most popular choice in Kaggle in 2019 [[ref](#)].
- [Approximate Nearest Neighbor \(ANN\) Benchmark](#)
- [Recommender Systems in Practice – Towards Data Science](#)
- [Association Rules – mlxtend](#)
- [Voting & Stacking – scikit-learn](#)

# Models of Models

# Overfitting

---



- A model **fits the training data too well**, and then **fails to predict**.
- Happens when the natural of the models, like trees, or over-tuning the hyperparameters.
- ← Green may be an **overfit**.
- Solutions:
  - *train\_score / test\_score* should be around 1.
  - **Test-Train Split**
  - **Cross Validation**

# Data Leakage

---



- The training data which **leads a high performance** is **not available when prediction**. Not the “leakage” in the security area.
- Two major types: [ref]
  - **Target Leakage:** like *treatment\_bool, disease\_bool* are all in the training set.
  - **Train-Test Contamination:** like backfilling train by test.
- Solutions:
  - Pipeline
  - Explanation  
(+ Domain Knowledge)



# Spurious Relationship

---

- The model **uses a false relationship to predict.**
- ← Get the 90% accuracy by “the background is snowy, so the animal is Husky.” [ref]
- Solution:
  - Explanation  
(+ Domain Knowledge)



# Model-Market Fit

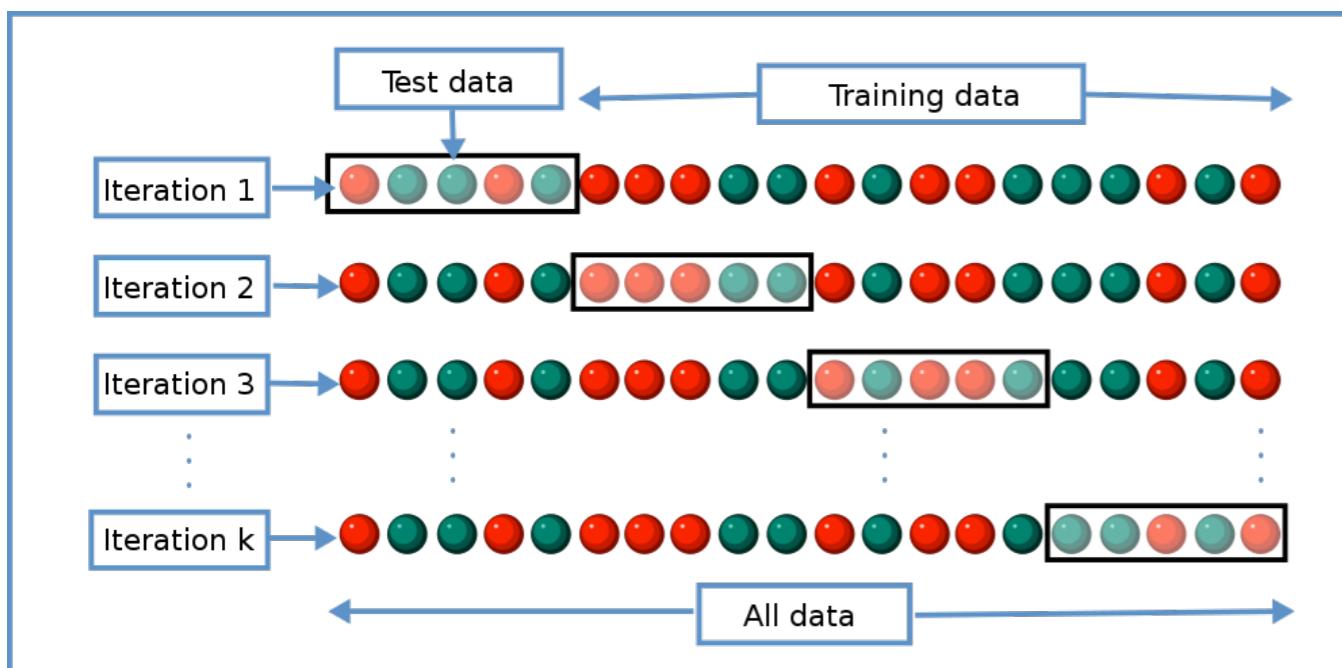
---

- “Product-Market Fit”.
- ← “Hey, this house is super similar to the one you just bought, buy one more?”
- “I build this model by ten years, please buy one!”
- Solution:
- Model Development

# Cross Validation

.....

- Test-Train Split is simple, but can't cover all the data.
  - Covers all the data by various strategies.
- ← K-Fold Cross Validation  
(K-Fold CV)





# Pipeline

---

- Consider we have the splits, but **cannot mix them**, so **fix** the processing steps to avoid.
- Handout: pipe\_and\_cv.ipynb.

## See Also

---

- [Cross validation iterators – scikit-learn](#)
  - Choose by the data generating process.
- [Exhaustive Grid Search – scikit-learn](#)
  - Search the best hyperparameters automatically.
- [AWS Data Pipeline](#)
  - Another common definition of the “pipeline”.



# Model Development

---

- “Software Development”
- How to “model-market fit”?
- Delight people with fast release!
- People must like your model:
  - Users.
  - Domain experts.
- Release faster; then learn faster, ideally 1–2 weeks.

## See Also

---

- The Analysis Steps
  - A suggested method to make an analysis, may be an analysis of a model or a data-driven product.
- The Study Designs
  - Besides the A/B testing, some not costly methods.
- The Mini-Scrum
  - How to work with a team efficiently.

# Temporal Data

---

- A Spurious Relationship happens naturally between independent non-stationary variables, like the mean varies respect to time.
- You may want a unit root test like the ADF test, or Domain Knowledge.
- The methods for temporal data:
  - tsa & statespace – statsmodels
  - Cross validation of time series data – scikit-learn

# Recap

---

- Exploratory like PCA helps to understand the data.
- Inference like statistical regressions finds the insights out.
- Preprocessing to feed easy-to-digest data to models.
- Inferences help to build better predictions.
- Delight people with fast release! 😊

# Image Credits

---

- “Linear PCA vs. Nonlinear Principal Manifolds”: [https://en.wikipedia.org/wiki/Principal\\_component\\_analysis#/media/File:Elmap\\_breastcancer\\_wiki.png](https://en.wikipedia.org/wiki/Principal_component_analysis#/media/File:Elmap_breastcancer_wiki.png)
- “SVM”: [https://en.wikipedia.org/wiki/Support\\_vector\\_machine#/media/File:SVM\\_margin.png](https://en.wikipedia.org/wiki/Support_vector_machine#/media/File:SVM_margin.png)
- “SVM With RBF Kernel”: [https://scikit-learn.org/stable/auto\\_examples/svm/plot\\_rbf\\_parameters.html](https://scikit-learn.org/stable/auto_examples/svm/plot_rbf_parameters.html)
- “Tree”: [https://en.wikipedia.org/wiki/Decision\\_tree\\_learning#/media/File:Cart\\_tree\\_kyphosis.png](https://en.wikipedia.org/wiki/Decision_tree_learning#/media/File:Cart_tree_kyphosis.png)
- “PCA vs. LDA”: [https://sebastianraschka.com/Articles/2014\\_python\\_lda.html](https://sebastianraschka.com/Articles/2014_python_lda.html)
- “Overfitting”: <https://en.wikipedia.org/wiki/Overfitting#/media/File:Overfitting.svg>
- “Data Leakage”: <https://www.kaggle.com/dansbecker/data-leakage>
- “Husky”: <https://en.wikipedia.org/wiki/Husky>
- “Wolf”: [https://en.wikipedia.org/wiki/Wolf#/media/File:Front\\_view\\_of\\_a\\_resting\\_Canis\\_lupus\\_ssp.jpg](https://en.wikipedia.org/wiki/Wolf#/media/File:Front_view_of_a_resting_Canis_lupus_ssp.jpg)
- “Houses”: <https://unsplash.com/photos/vZEPXDQHR4s>
- “K-Fold Cross-Validation”: [https://en.wikipedia.org/wiki/Cross-validation\\_\(statistics\)#/media/File:K-fold\\_cross\\_validation\\_EN.svg](https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.svg)
- “Pipeline”: <https://unsplash.com/photos/KP6XQIEjjPA>
- “Smile”: <https://unsplash.com/photos/g1Kr4Ozfoac>