

Introduction to Scikit-learn

a Python Machine Learning Package



Titli Sarkar
Computational Scientist
Titli.Sarkar@nih.gov

Advanced Biomedical & Computational Science (ABCS)



Features	Python	R
Purpose of Existence	<ul style="list-style-type: none"> <input type="checkbox"/> Open-source <input type="checkbox"/> Beginner-friendly <input type="checkbox"/> Multi-Purpose <input type="checkbox"/> Scalable 	<ul style="list-style-type: none"> <input type="checkbox"/> Open source <input type="checkbox"/> Best for data visualization <input type="checkbox"/> Complex statistical analysis <input type="checkbox"/> Data analysis tasks
Usability	<ul style="list-style-type: none"> <input type="checkbox"/> Simplicity and code readability <input type="checkbox"/> Smooth learning curve 	<ul style="list-style-type: none"> <input type="checkbox"/> Steep learning curve for developers who do not have prior statistical language programming skills.
Performance	<ul style="list-style-type: none"> <input type="checkbox"/> Faster for computationally intensive tasks and large-scale data processing, especially when optimized. <input type="checkbox"/> Better memory usage for large datasets. 	<ul style="list-style-type: none"> <input type="checkbox"/> Relatively slower than python for large datasets and complex operations. <input type="checkbox"/> Optimization function like <i>lapply()</i> can improve performance.
Data Manipulation	<ul style="list-style-type: none"> <input type="checkbox"/> <i>Pandas</i> -> data cleaning, transformation, and manipulation (Table as Dataframe). 	<ul style="list-style-type: none"> <input type="checkbox"/> Excels with data frames and offers various packages for data reshaping and manipulation, often favored by <i>statisticians</i>.
Data Visualization	<ul style="list-style-type: none"> <input type="checkbox"/> <i>Matplotlib, Seaborn, and Plotly</i> -> extensive visualization options. 	<ul style="list-style-type: none"> <input type="checkbox"/> <i>ggplot2</i> -> widely-used powerful library for creating high-quality, customizable statistical graphics.
Statistical Analysis	<ul style="list-style-type: none"> <input type="checkbox"/> <i>NumPy, SciPy, and StatsModels</i> for numerical and statistical computing, with a strong focus on machine learning. 	<ul style="list-style-type: none"> <input type="checkbox"/> Designed specifically for statistical computing, with a vast array of built-in statistical functions and specialized packages for various analyses.
Machine Learning	<ul style="list-style-type: none"> <input type="checkbox"/> Dominant in machine learning and deep learning with libraries like <i>scikit-learn, TensorFlow, and PyTorch</i>. 	<ul style="list-style-type: none"> <input type="checkbox"/> Provides libraries like <i>Caret and MLR</i> for machine learning, though its ecosystem is generally more focused on traditional statistical modeling.

What should I choose: R or Python?

Summary:

- **R** is often preferred by statisticians and researchers for its specialized statistical capabilities and exceptional data visualization tools (especially ggplot2).
- **Python** is a more versatile, general-purpose language favored for its ease of learning, extensive machine learning libraries, and strong capabilities in data engineering and deployment.
- The choice between Python and R often depends on the specific project requirements, team expertise, and industry standards. Many data professionals find value in learning both languages to leverage their respective strengths.

Why Scikit-learn?

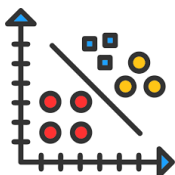
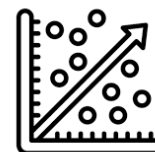
What is Scikit-learn?

- It is an *open-source* Machine Learning library for the Python programming language.
- It provides a wide range of **built-in** metrics for algorithms and tools for various machine learning tasks, including:



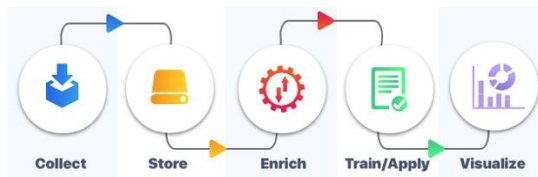
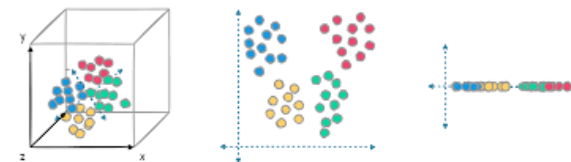
Classification: Identifying which category an object belongs to (e.g., spam detection, image recognition).

Regression: Predicting a continuous-valued attribute associated with an object (e.g., stock prices, drug response).



Clustering: Automatic grouping of similar objects into sets (e.g., customer segmentation).

Dimensionality Reduction: Reducing the number of random variables to consider (e.g., for visualization or increased efficiency).



Model Selection: Comparing, validating, and choosing parameters and models (e.g., grid search, cross-validation).

Preprocessing: Feature extraction and normalization (e.g., transforming input data for use with machine learning algorithms).



REMOVING NOISE



STANDARDIZATION



COMBINING SOURCES



SIMPLIFICATION



HANDLING MISSING VALUES

Built on top of other popular Python libraries like *NumPy*, *SciPy*, and *Matplotlib*, scikit-learn offers a user-friendly and consistent API, making it a popular choice for both beginners and experienced practitioners in the field of machine learning and data science.

Scikit-learn vs. other Python ML libraries



scikit-learn vs. Pandas

- **Pandas** is ideal for data manipulation and analysis.
- **Scikit-learn** is perfect for machine learning tasks.
- In practice, these libraries often work together – NumPy for calculations, Pandas for data preparation, and Scikit-learn for model building.

Pandas Basic Operations

Pandas simplifies data handling by enabling efficient preprocessing, cleaning, transformation, and visualization.



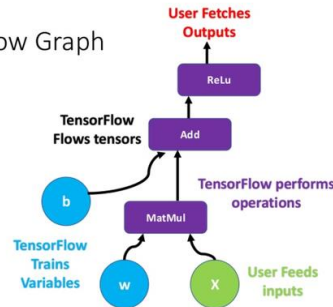
9/18/2025



scikit-learn vs. TensorFlow

- **Scikit-Learn** is generally considered *better* for beginners due to its simplicity and ease of use.
- **TensorFlow** has a steeper learning curve and is more suitable for individuals with prior experience or those specifically interested in deep learning.

TensorFlow Graph



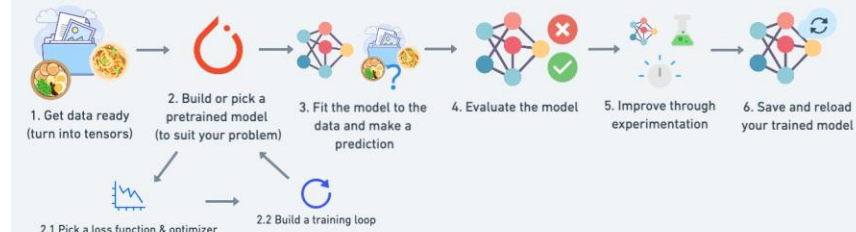
Intro to scikit-learn



scikit-Learn vs. PyTorch

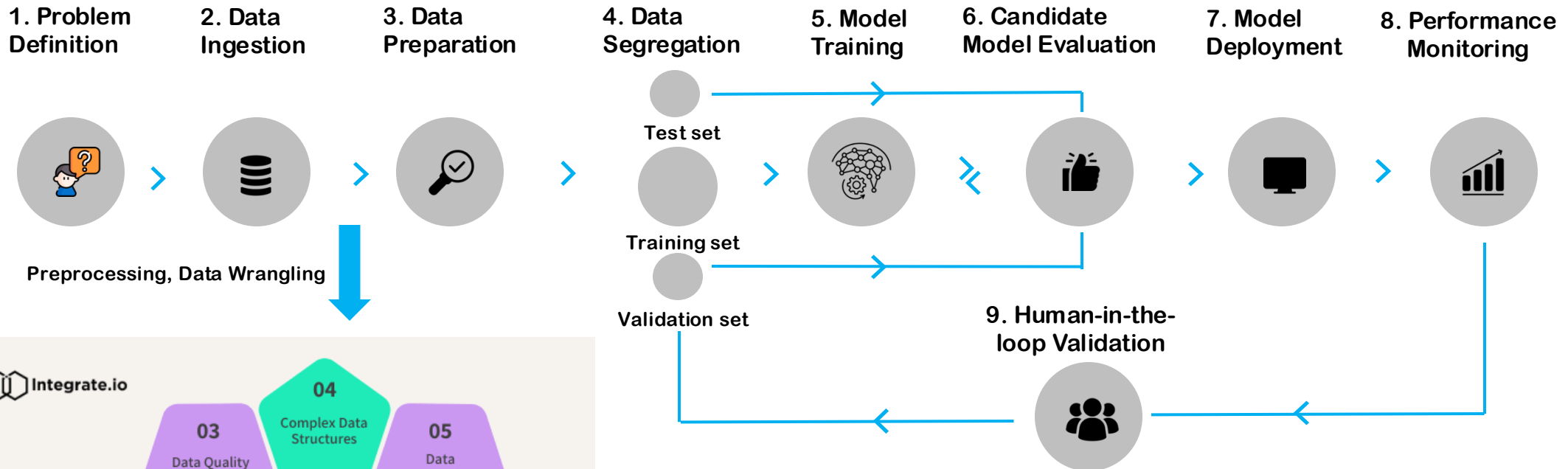
- **Sklearn** is built on top of Python libraries like NumPy, SciPy, and Matplotlib, and is simple and efficient for data analysis, efficient for machine learning.
- **PyTorch** is designed for deep learning projects, large-scale applications, custom neural network architectures, and when leveraging GPU acceleration is necessary.

A PyTorch Workflow



4

Basic Machine Learning pipeline

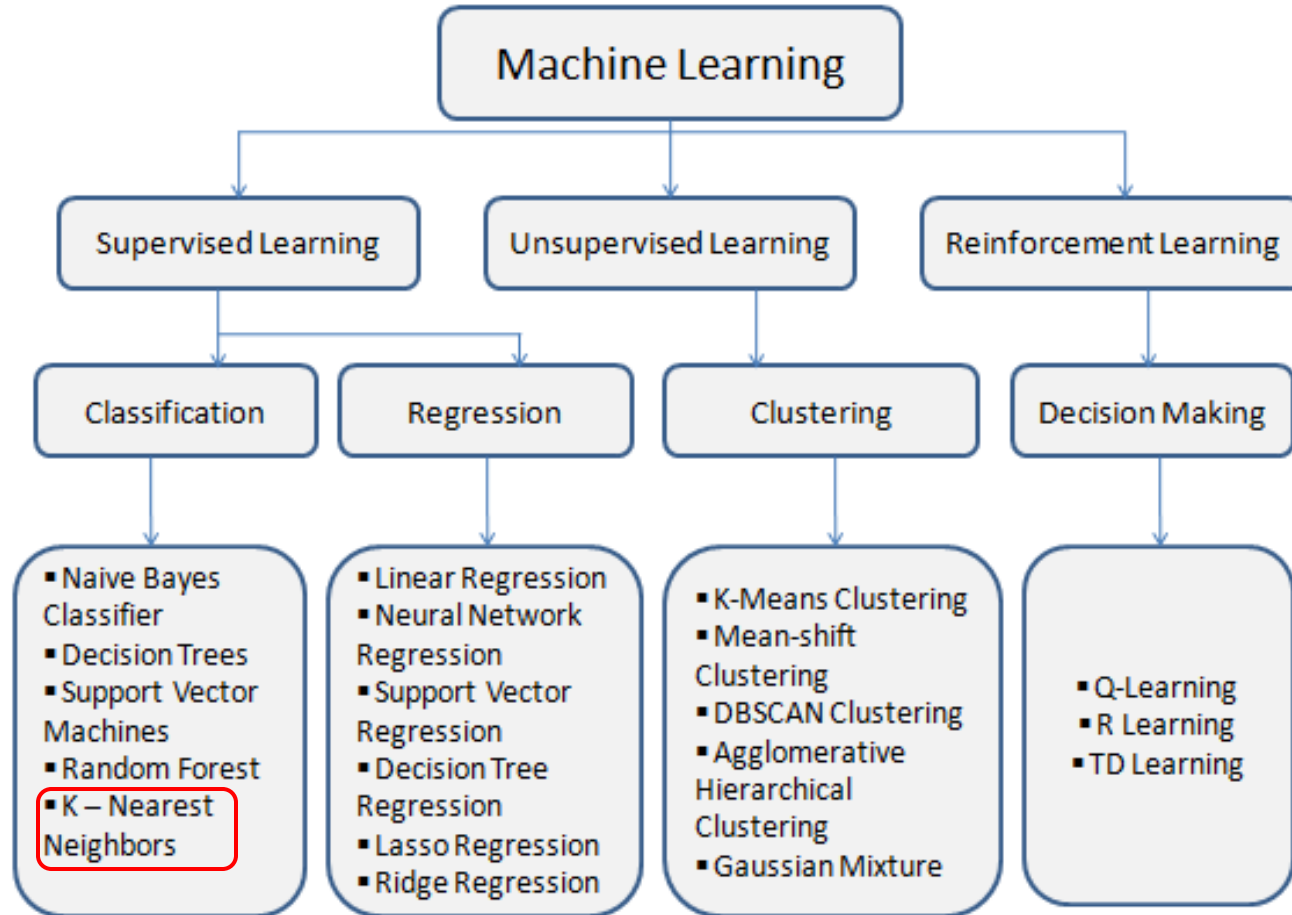


Machine Learning Model Training:

Iterative Process: an algorithm learns from data to make predictions or perform specific tasks.

- First, choose an appropriate algorithm.
- Second, feed the algorithm a dataset (model training),
- Next, through iterative adjustments to its internal parameters (hyperparameter tuning) , refine the algorithm's ability to recognize patterns and make accurate predictions (minimize loss).

Machine Learning Model selection



Useful Resources:

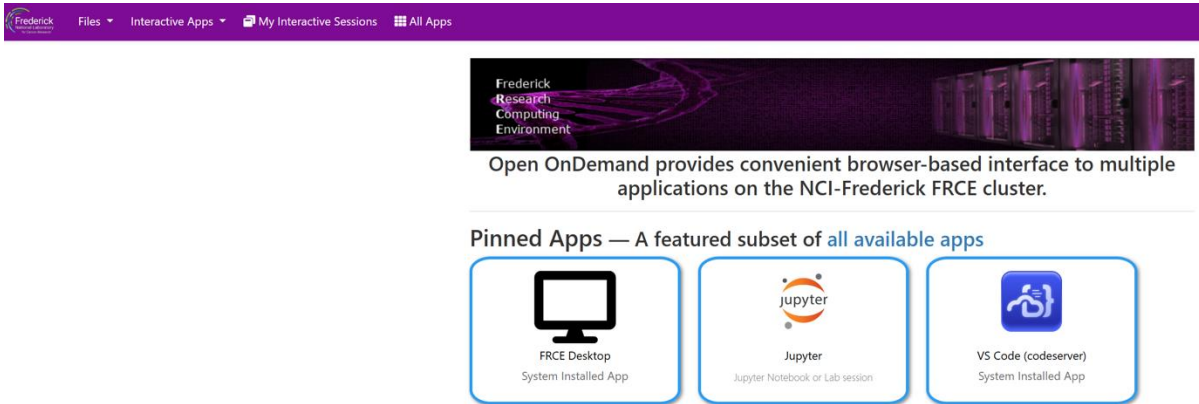
- [Getting Started with HPC OnDemand: A New Interface for NIH HPC Users:](#) Jonathan Goodson (CIT)
- [BTEP Coding Club: Submitting Scripts to the Biowulf Batch System:](#) Joe Wu (BTEP)
- [Documenting Data Analysis with Jupyter Lab:](#) Joe Wu (BTEP)
- [https://bioinformatics.ccr.cancer.gov/btep/resources/scientific-software/:](https://bioinformatics.ccr.cancer.gov/btep/resources/scientific-software/) BTEP softwares
- [Decision Trees, Survival Trees, and Random Forest:](#) Brian Luke (ABCS)
- [Decision Trees, Survival Trees, and Random Forest: Practical Examples with R Programming:](#) Brian Luke (ABCS)
- [Correlation and Regression Methods in Statistical Analysis:](#) Alexander Mitrophanov (DMS)
- [Introduction to Clustering :](#) Brian Luke (ABCS)
- [Clustering with R and Rstudio_:](#) Brian Luke (ABCS)
- [Survival Analysis in R Using the survival Package:](#) Duncan Donohue (DMS)
- [Introduction to Data Exploration:](#) Duncan Donohue (DMS)
- [Missing Values and Data Transformations:](#) Duncan Donohue (DMS)

How can I get a hands-on?

Option 1: Download and install Anaconda3 to your computer (large space), which comes with Python, Spider (run .py scripts) and Jupyter (run .ipynb notebooks). **Not Recommended** or, save yourself from the headache of installation!

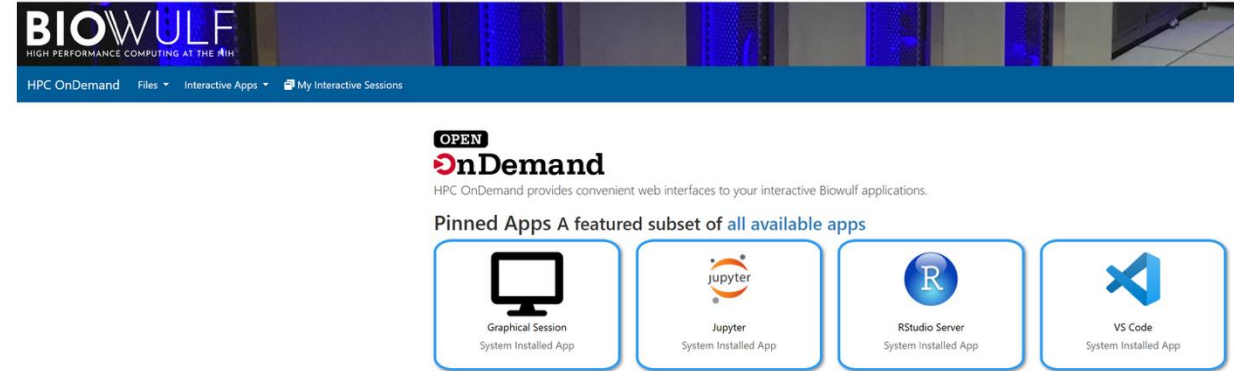
Option 2: NIH HPC have all of it installed and readily available for users at both Biowulf and FRCE via Open Ondemand.

- <https://ondemand.ncifcrf.gov>



FRCE

- <https://hpcondemand.nih.gov/>



Biowulf

***Important!**

1. You need to navigate to the directory where your data and code are located. Jupyter will start from the directory you selected at the startup.
2. Please be respectful to others and be mindful while requesting for resources, Do NOT ask for resources you don't need.
 - Jupyter notebook is an interactive way of Python scripting, mostly used for learning purpose. Almost never you need a GPU node for running a Jupyter notebook, so, please do not ask.
 - Also, be careful about amount of memory and CPU cores.
3. Do NOT forget to relinquish resources after you are done.
 - You need to come back to you session homepage and 'DELETE' the session once you are finished.

Dataset

- <https://www.kaggle.com/datasets/iammustafatz/diabetes-prediction-dataset> or,
- https://www.kaggle.com/code/mahmoudbahnasy29/diabetes?select=diabetes_prediction_dataset.csv
- **diabetes_prediction_dataset.csv** file: contains medical and demographic data of patients along with their diabetes status, whether positive or negative. It consists of various features such as age, gender, body mass index (BMI), hypertension, heart disease, smoking history, HbA1c level, and blood glucose level. The Dataset can be utilized to construct machine learning models that can predict the likelihood of diabetes in patients based on their medical history and demographic details.

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

100000 rows × 9 columns

Information about the Data

```
df.info(verbose=True)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
#   Column                Non-Null Count  Dtype
---  ---                ---
0   gender                100000 non-null object
1   age                   100000 non-null float64
2   hypertension          100000 non-null int64
3   heart_disease         100000 non-null int64
4   smoking_history       100000 non-null object
5   bmi                   100000 non-null float64
6   HbA1c_level           100000 non-null float64
7   blood_glucose_level   100000 non-null int64
8   diabetes              100000 non-null int64
dtypes: float64(3), int64(4), object(2)
memory usage: 6.9+ MB
```

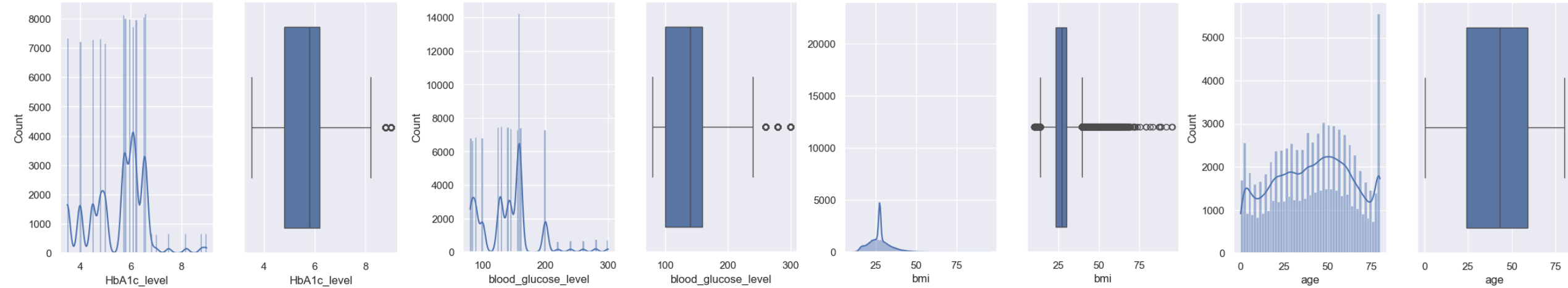
Data Statistics

```
df.describe(include="all").T
```

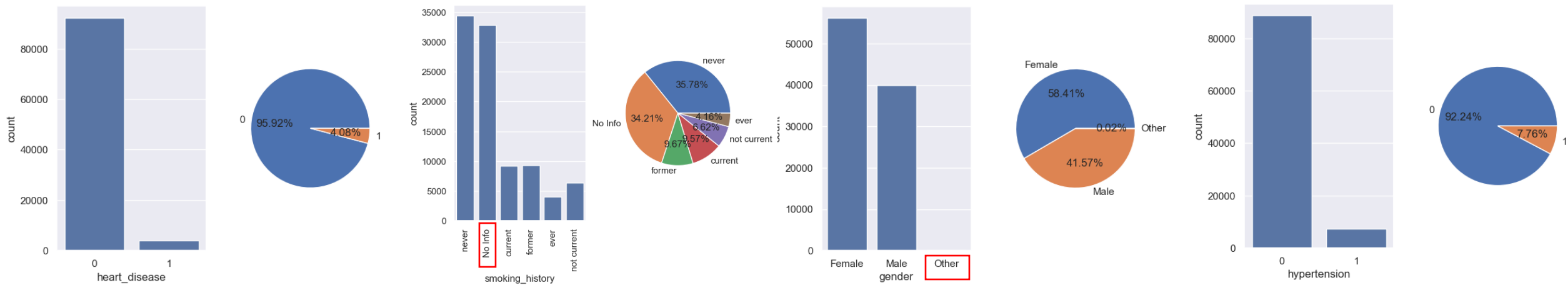
	count	mean	std	min	25%	50%	75%	max
gender	100000.0	1.585340	0.493031	0.00	1.00	2.00	2.00	2.00
age	100000.0	41.885856	22.516840	0.08	24.00	43.00	60.00	80.00
hypertension	100000.0	0.074850	0.263150	0.00	0.00	0.00	0.00	1.00
heart_disease	100000.0	0.039420	0.194593	0.00	0.00	0.00	0.00	1.00
smoking_history	100000.0	1.433770	1.627925	0.00	0.00	1.00	3.00	5.00
bmi	100000.0	27.320767	6.636783	10.01	23.63	27.32	29.58	95.69
HbA1c_level	100000.0	5.527507	1.070672	3.50	4.80	5.80	6.20	9.00
blood_glucose_level	100000.0	138.058060	40.708136	80.00	100.00	140.00	159.00	300.00
diabetes	100000.0	0.085000	0.278883	0.00	0.00	0.00	0.00	1.00

Univariate Analysis

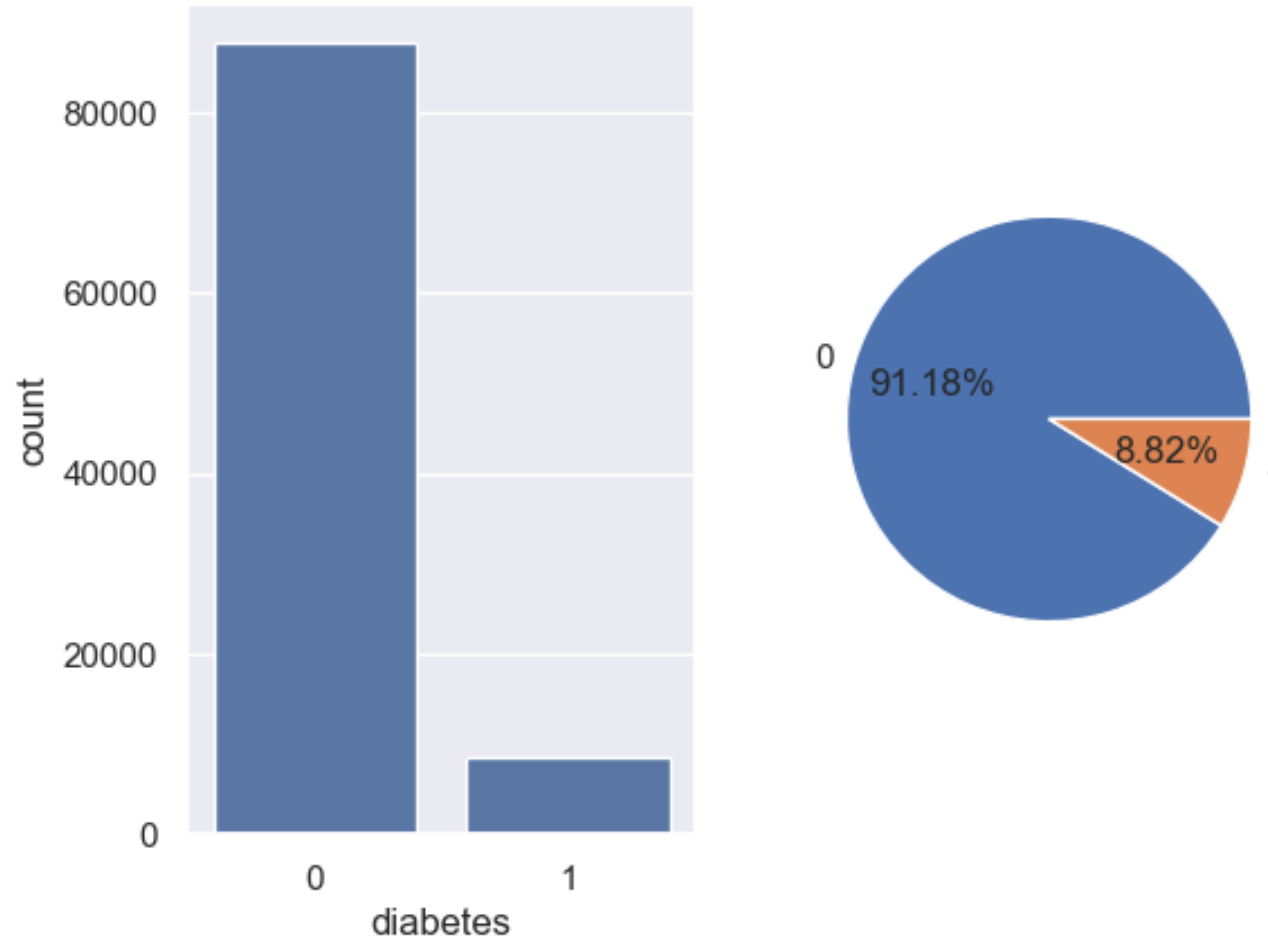
First, let's take a look at the data visually ...



Pie plot for categorical variables



Target Variable 'diabetes' shows Data Imbalance



Bivariate Analysis

- Check co-relation between features
- hue =diabetes shows percentage of diabetes positive or negative patients within each feature variable



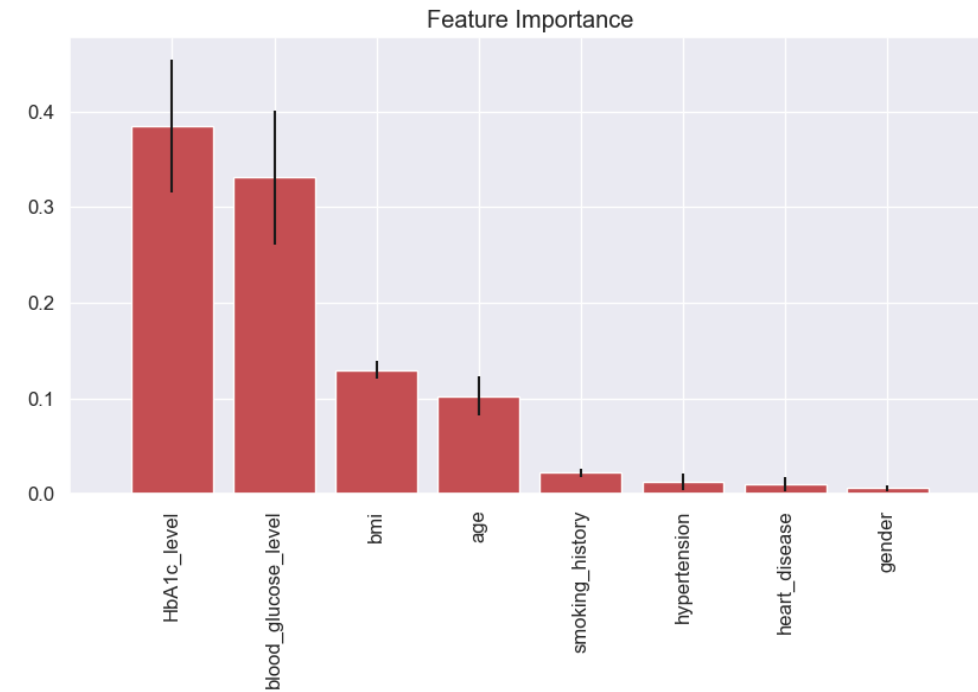
diabetes
• 0
• 1

9/18/2025

Intro to scikit-learn

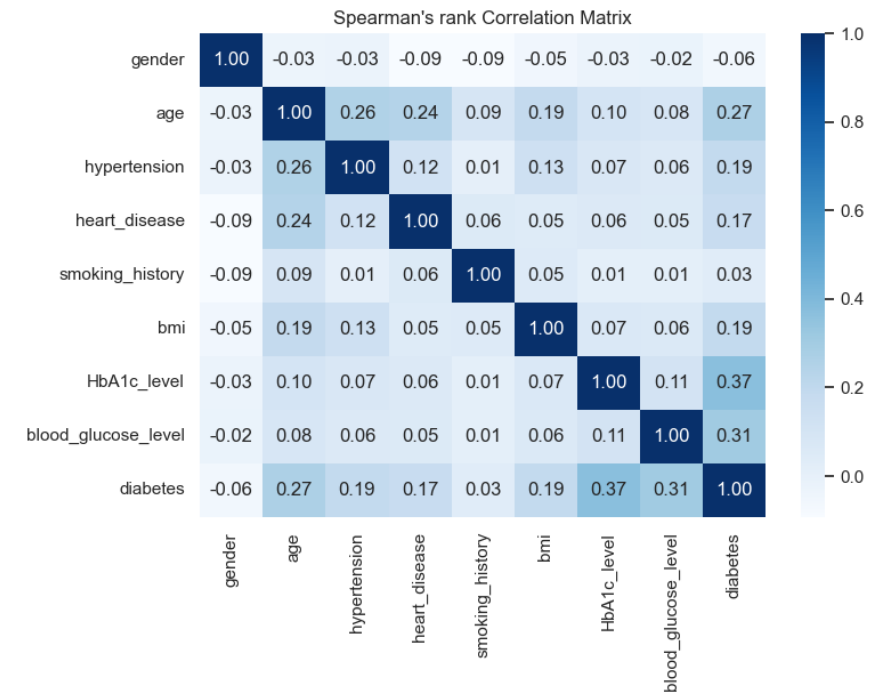
Preprocessing

- 3854 duplicates -> drop them, keeping the first.
- No null values. Consider zeros for some columns as null and remove them.
- Standardization (min-max Normalization or Scaling) to rescales features between 0 and 1.
- Distance-based Machine Learning algorithms can only read numerical values, therefore, convert categorical columns to numeric.



```
df_min_max_scaled.describe(include="all").T
```

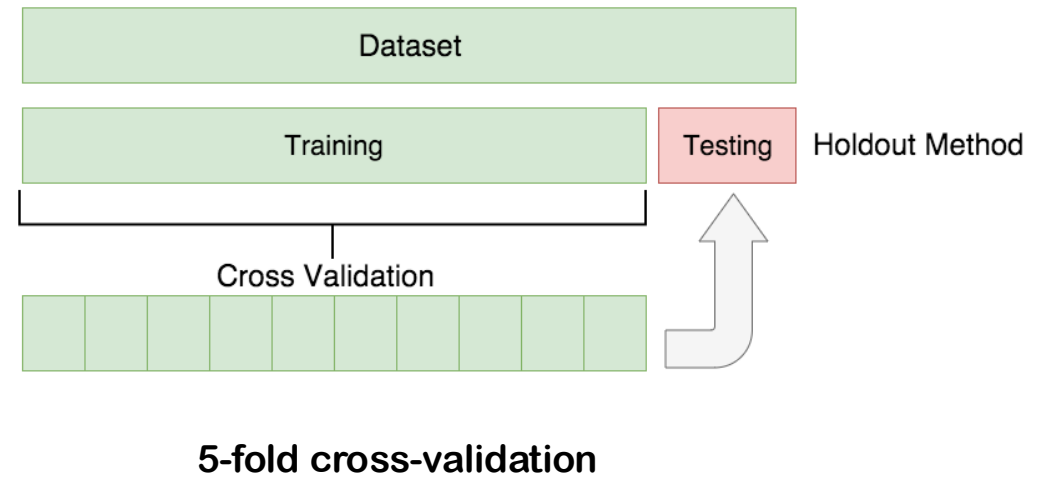
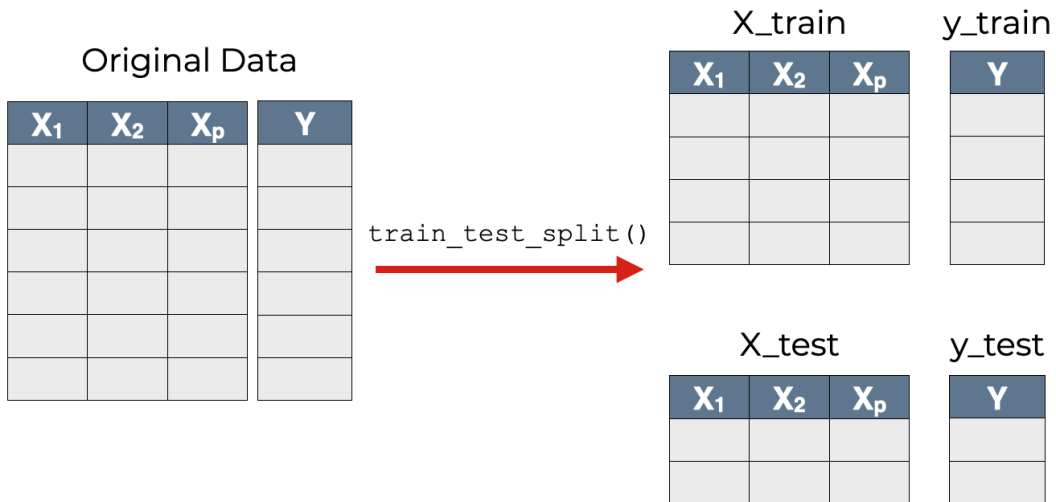
	count	mean	std	min	25%	50%	75%	max
gender	100000.0	0.792670	0.246515	0.0	0.500000	1.000000	1.000000	1.0
age	100000.0	0.523096	0.281742	0.0	0.299299	0.537037	0.749750	1.0
hypertension	100000.0	0.074850	0.263150	0.0	0.000000	0.000000	0.000000	1.0
heart_disease	100000.0	0.039420	0.194593	0.0	0.000000	0.000000	0.000000	1.0
smoking_history	100000.0	0.286754	0.325585	0.0	0.000000	0.200000	0.600000	1.0
bmi	100000.0	0.202040	0.077460	0.0	0.158964	0.202031	0.228408	1.0
HbA1c_level	100000.0	0.368638	0.194668	0.0	0.236364	0.418182	0.490909	1.0
blood_glucose_level	100000.0	0.263900	0.185037	0.0	0.090909	0.272727	0.359091	1.0
diabetes	100000.0	0.085000	0.278883	0.0	0.000000	0.000000	0.000000	1.0



Data splitting

```
# split data using scikit-Learn
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 1/3, random_state = 42)
```



Fundamental step to ensure that the model's performance is accurately evaluated and to prevent **overfitting** (model performs well on the training data, but poorly performs on unseen test data).

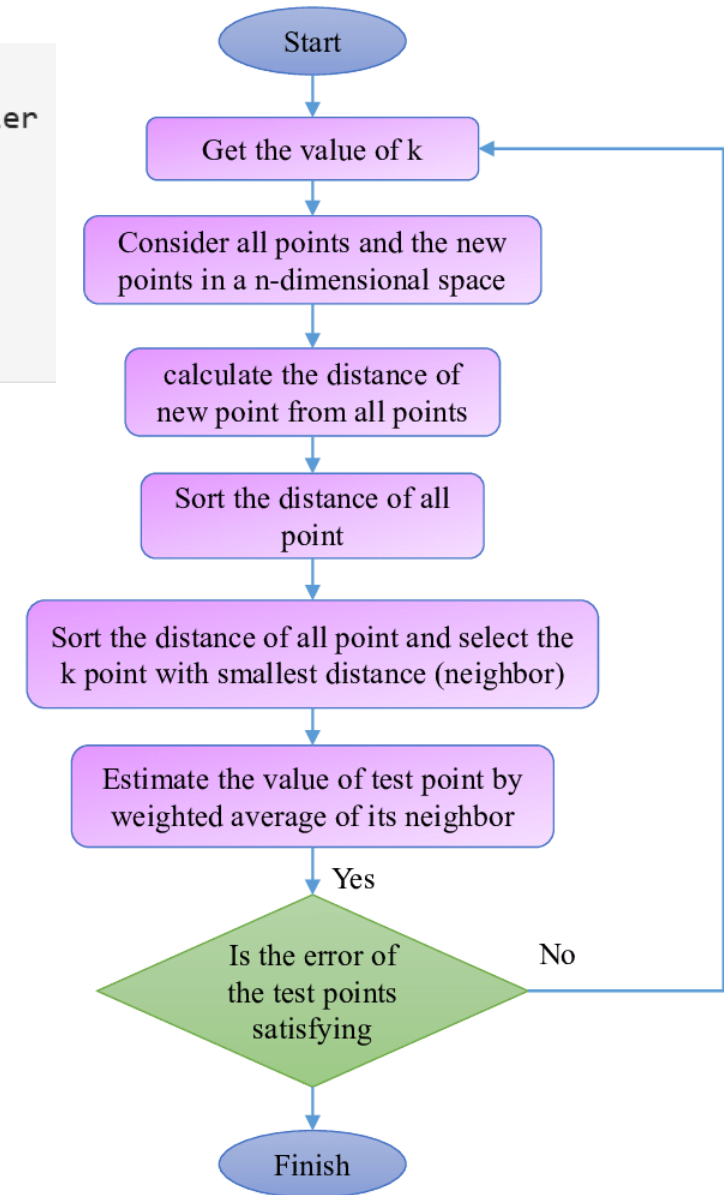
Classification model : k Nearest Neighbors (knn)

```
# use in-built knn ML model from scikit-learn
from sklearn.neighbors import KNeighborsClassifier

k = best_k
model_knn = KNeighborsClassifier(n_neighbors=k)
model_knn.fit(X_train, y_train)
pred_knn = model_knn.predict(X_test)
```

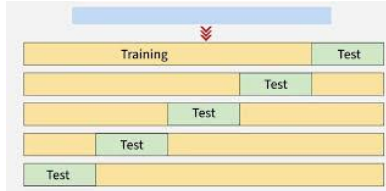
Key Aspects:

- **Supervised** ML algorithm,
- Used for both Classification and Regression
- **Non-parametric**: It makes no assumptions about the underlying data distribution.
- **Distance-based**: Its performance heavily relies on the chosen distance metric.
- *** Sensitive to 'k'**: The choice of 'k' is crucial; a small 'k' can lead to overfitting, while a large 'k' can lead to underfitting. Cross-validation is often used to find an optimal 'k'.
- **Computational cost**: For large datasets, calculating distances to all training points can be computationally expensive during prediction.
- Considered as a "lazy" learning algorithm because it does not build a model during the training phase but instead stores the entire training dataset. All computation is deferred until a new data point needs to be classified or its value predicted.



How to choose the value of k ?

- Crucial step for achieving good model performance. While there's NO single "best" method, several approaches are commonly employed.



Cross-Validation

Most robust method.

Iterate through a range of k values (e.g., from 1 to a reasonable upper limit):

- For each k , perform ***k-fold cross-validation*** on your training data.
- Calculate the average performance metric (e.g., ***accuracy, F1-score***) for each k .
- Select the k that yields the best average performance.

* Important Considerations:

- **Odd k for Classification:** When dealing with binary or multi-class classification, choosing an **odd value** for k is generally preferred to prevent ties when determining the majority class among the neighbors.
- **Computational Cost:** Larger k values increase the computational cost of finding neighbors.
- **Bias-Variance Trade-off:** A small k can lead to high variance (overfitting to noise), while a large k can lead to high bias (*oversmoothing and underfitting*). The goal is to find a k that balances this trade-off.

Domain Knowledge

Consider the nature of your data and problem.

If you expect local patterns to be very distinct, a smaller k might be appropriate.

If the data is noisy, a larger k can help to smooth out the decision boundaries.

Model Evaluation

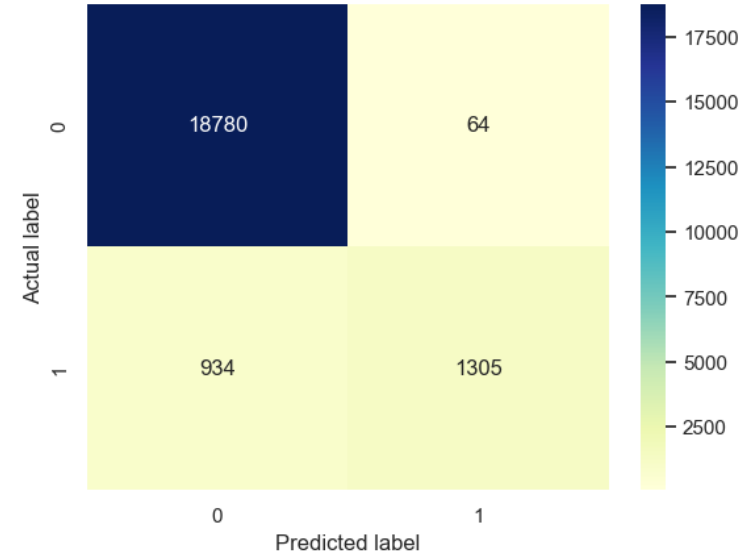
```
# Evaluate the model using scikit-Learn metrics
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score
from sklearn.metrics import mean_squared_error, r2_score, roc_auc_score
from sklearn.metrics import confusion_matrix, classification_report

pred_knn = model_knn.predict(X_test)

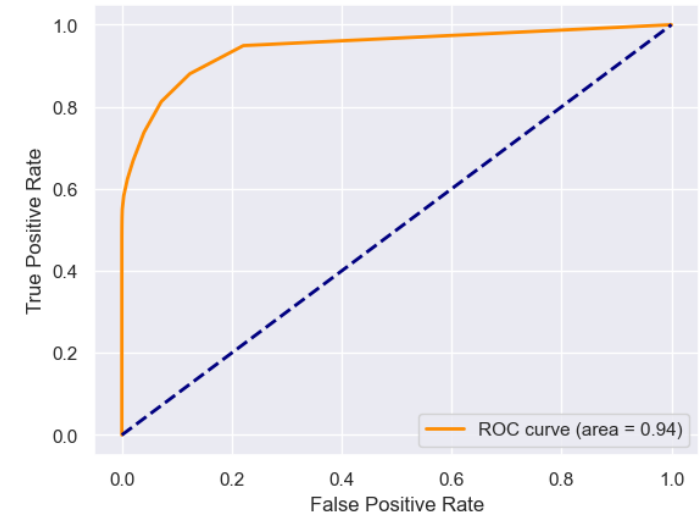
accuracy = accuracy_score(y_test, pred_knn)
precision = precision_score(y_test, pred_knn)
recall = recall_score(y_test, pred_knn)
auc = roc_auc_score(y_test, pred_knn)
cr = classification_report(y_test, pred_knn)
cm = confusion_matrix(y_test, pred_knn)
```

	precision	recall	f1-score	support
0.0	0.95	1.00	0.97	18844
1.0	0.95	0.58	0.72	2239
accuracy			0.95	21083
macro avg	0.95	0.79	0.85	21083
weighted avg	0.95	0.95	0.95	21083

Confusion matrix



ROC curve



What are the disadvantages of scikit-learn?

- hyperparameters and the search space of the models are awkwardly defined.
- think of built-in hyperparameter spaces and AutoML algorithms.
- with scikit-learn, a pipeline step can only have some hyperparameters, but they don't each have an hyperparameter distribution.

Acknowledgement

ABCS

Uma Mudunuri, PhD

Brian Luke, PhD

Joseph Ivanic, PhD

Wojciech Kasprzak, MS

David Bell, PhD

Ernesto Suarez Alvarez, PhD

BTEP

Amy Stonelake, PhD

Alexandra Emmons, PhD

Questions?