2021 Spring – Introduction to Machine Learning
Final Presentation

Titanic - Machine Learning from Disaster

Presenter:

102012805 Mey Yeh, 107011153 Huey-Chii Liang, 109032805 Chih-Mei Young

Instructor: Prof. Shun-Chi Wu

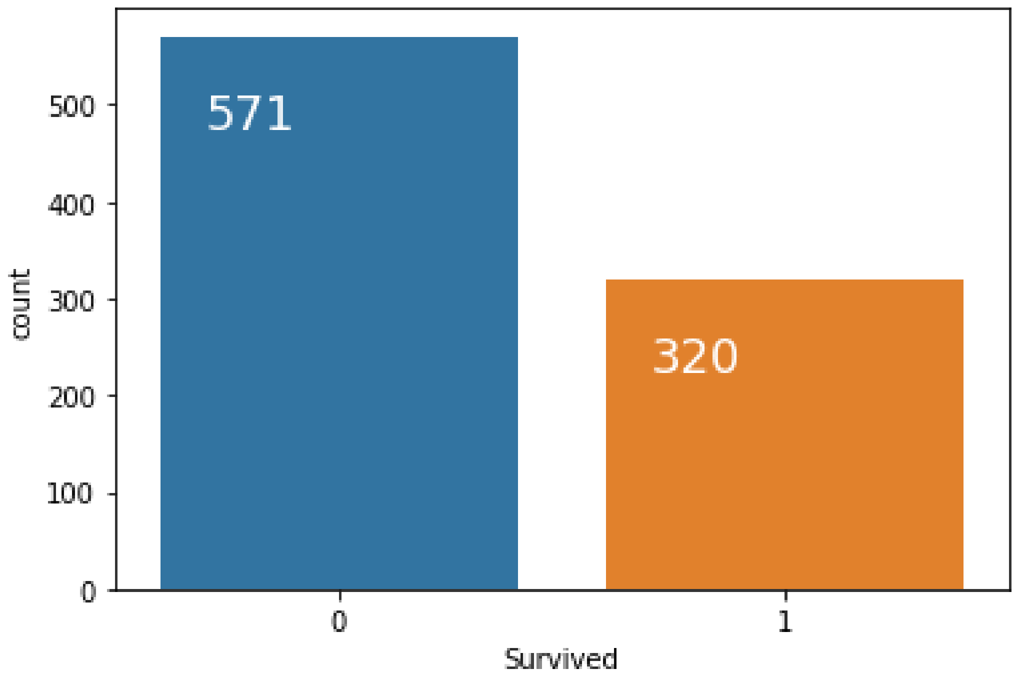
Table of Contents

Basic Information and Data Interpretation

■ Training data information:

- Dataset.info

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       758 non-null    float64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          702 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         890 non-null    float64
 10  Cabin        201 non-null    object
 11  Embarked     891 non-null    object
dtypes: float64(3), int64(4), object(5)
memory usage: 83.7+ KB
```



10 features

Missing Data

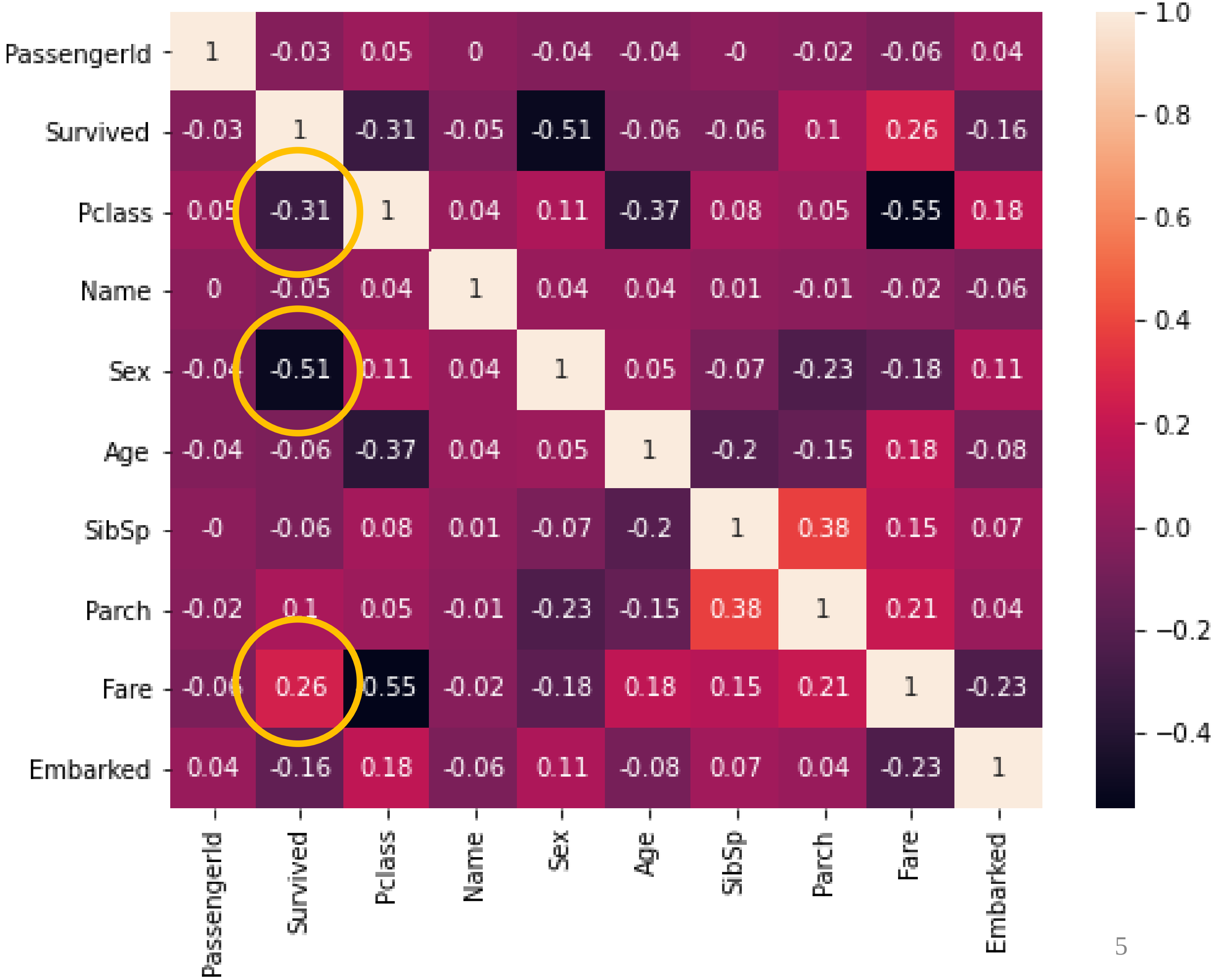
print(pd.isnull(train).sum())

	Total	%
~~Cabin~~	~~690~~	~~77.4~~
Age	189	21.2
Pclass	133	14.9
Fare	1	0.1
Embarked	0	0.0
~~Ticket~~	~~0~~	~~0.0~~
Parch	0	0.0
SibSp	0	0.0
Sex	0	0.0
Name	0	0.0
Survived	0	0.0
PassengerId	0	0.0

- Fill in rational value
  - Age (mean value)
  - Fare (mean value)
  - Pclass (observed)
- Delete the feature
  - Cabin, Ticket

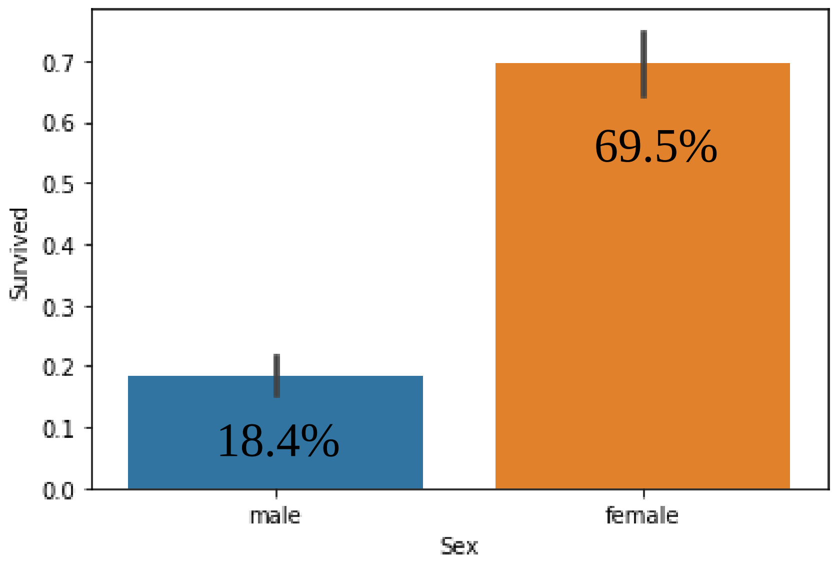
Correlation between Features

- **Seaborn:**
  - Correlation: Sex > Pclass > Fare

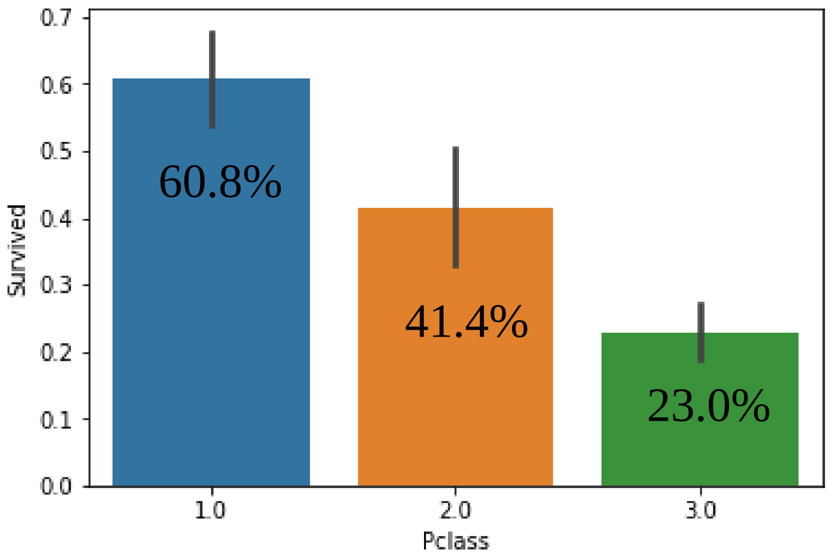


Visualize the dataset for selecting features-1

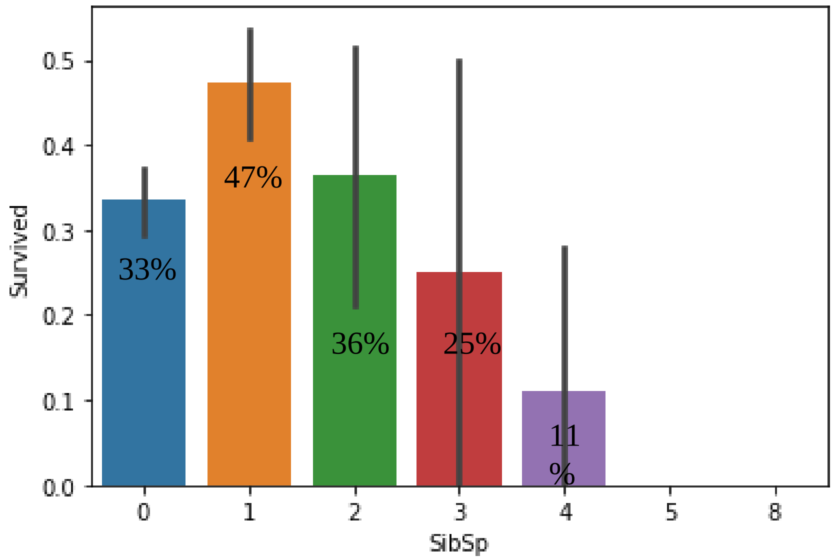
- Gender



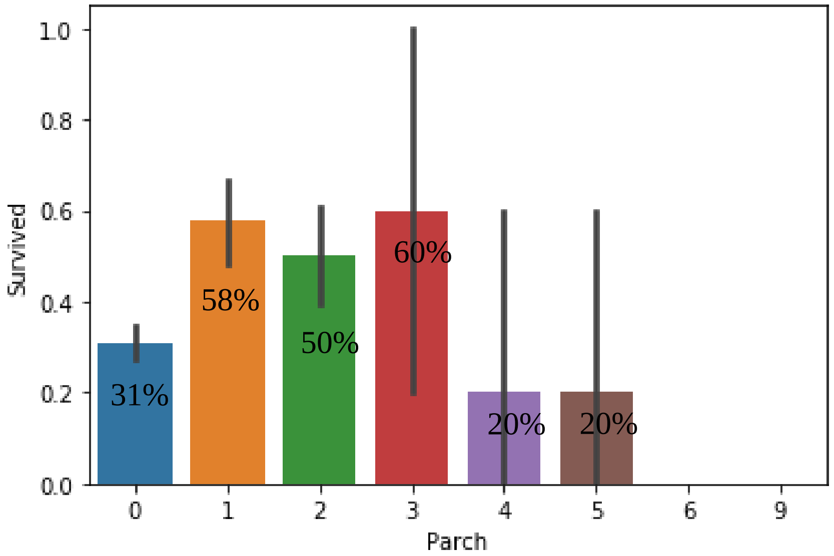
- Pclass



- Sibling

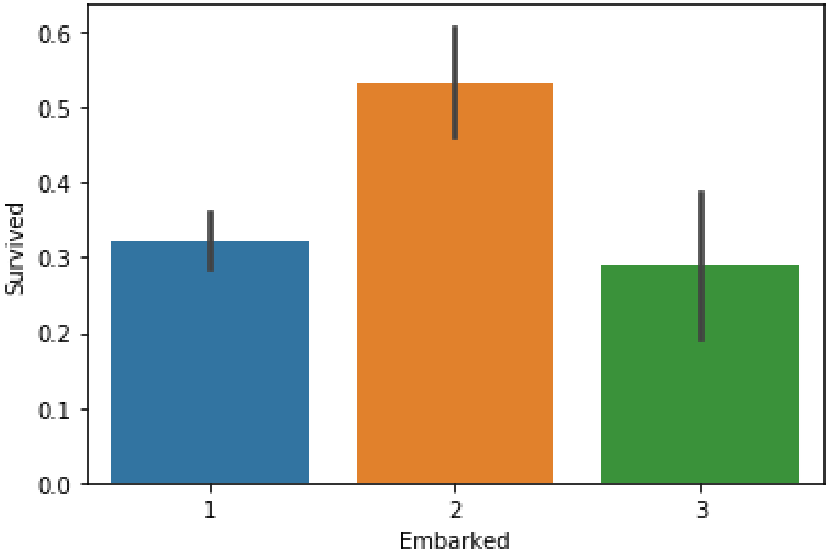


- Parch

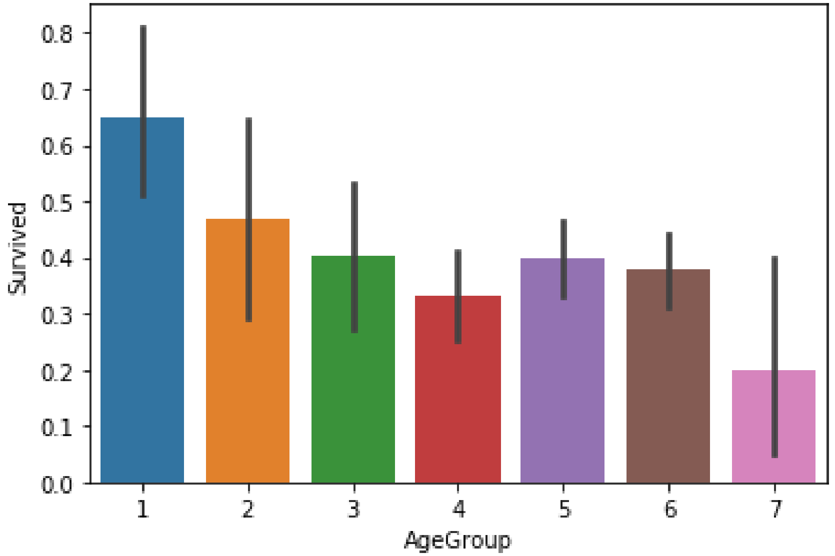


Visualize the dataset for selecting features-2

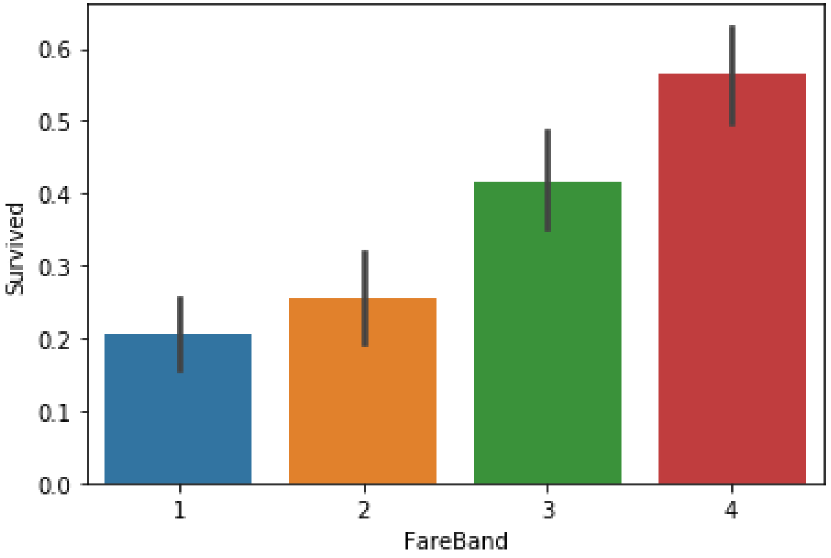
- Embarked



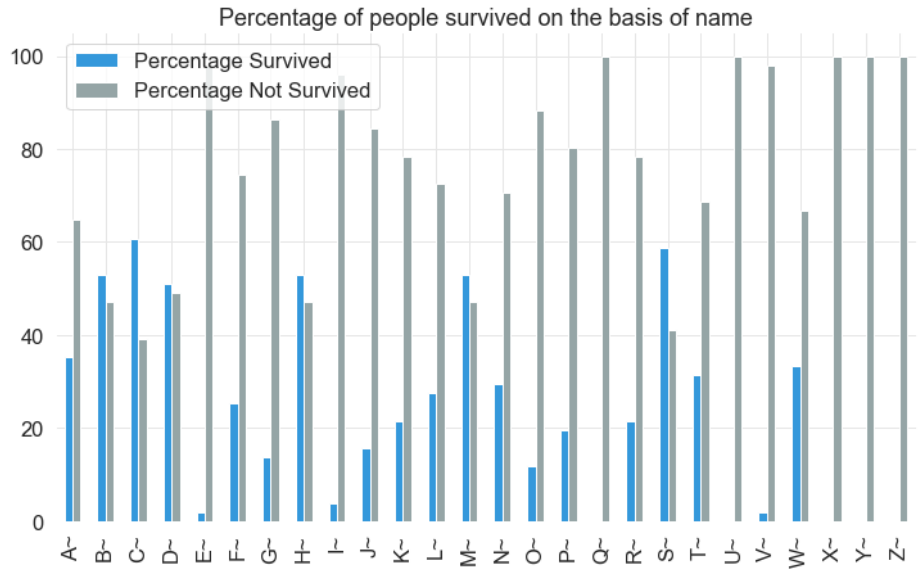
- Age



- Fare



- Name



Features Combination

Features	A	B	C	D	E	F
Pclass	observed	observed	observed	observed	Depend on Fare	Depend on Fare
Name	A->1, B->2 …	A->1, B->2 …	-	-	-	-
Sex	Male=0 Female=1	Male=0 Female=1	Male=0 Female=1	Male=0 Female=1	Male=0 Female=1	Male=0 Female=1
Age	Mean age	Random number (mean +std) Age group(1-6)	Random number (mean +std) Age group(1-6)	Random number (mean +std) Age group(1-6)	Random number (mean +std) Age group(1-6)	Random number (mean +std) Age group(1-6)
SibSp	+	-	-	+	+	+
Parch	+	-	-	+	+	+
Relatives	-	+	+	+	+	-
Ticket	-	-	-	-	-	-
Fare	-	-	Fare band (4)	Fare band (4)	Fare per person	Fare per person
Cabin	-	-	-	-	-	-
Embarked	S:1, C:2, Q:3	S:1, C:2, Q:3	S:1, C:2, Q:3	S:1, C:2, Q:3	S:1, C:2, Q:3	S:1, C:2, Q:3
Not alone	-	+	+	+	+	-

Selection of Models – Gaussian NBC

Gaussian Naïve Bayes Classifier

- Algorithm
  - A collection of classification algorithms based on Bayes' Theorem
  - Assumptions → Each feature makes an **equal** and **independent** contribution to the outcome
  - **Bayes' Theorem**: $P(y|X) = \frac{P(X|y)P(y)}{P(X)}$

    where y = class variable, $X = (x_1, x_2, x_3, x_4, \ldots, x_n)$ a dependent feature vector (of size n)
  - **Naïve Assumption**: $P(A,B) = P(A)P(B)$
  - Combination: $\boxed{P(y|x_1, x_2, x_3, x_4, \ldots, x_n) = \frac{P(x_1|y)P(x_2|y)P(x_3|y)\ldots P(x_n|y)\,P(y)}{P(x_1)P(x_2)P(x_3)\ldots P(x_n)}}$
  - **Gaussian** NBC:

    Continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution

    The likelihood of the features is assumed to be Gaussian, hence, conditional probability is given by

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}}\exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

Selection of Models – Gaussian NBC

- Code from scratch

```
#%% Gaussian NB
def split_classes(X_train, y_train):
    C1_count = (y_train == 1).sum()
    C2_count = (y_train == 0).sum()
    X_train_C1 = np.zeros([C1_count, X_train.shape[1]])
    X_train_C2 = np.zeros([C2_count, X_train.shape[1]])
    y_train_C1 = np.zeros([C1_count, 1])
    y_train_C2 = np.zeros([C2_count, 1])
    c1 = 0
    c2 = 0
    for i in range(len(y_train)):
        if y_train[i] == 1.:
            X_train_C1[c1] = X_train[i]
            y_train_C1[c1] = y_train[i]
            c1 = c1 + 1
        else:
            X_train_C2[c2] = X_train[i]
            y_train_C2[c2] = y_train[i]
            c2 = c2 + 1
    return X_train_C1, X_train_C2, y_train_C1, y_train_C2

def Gaussian_NB(X_train, y_train, X_test):
    y_pred = np.zeros(len(X_test))
    X_train_C1, X_train_C2, y_train_C1, y_train_C2 = split_classes(X_train, y_train)
    C1_mean = np.mean(X_train_C1, axis=0)
    C2_mean = np.mean(X_train_C2, axis=0)
    C1_std = np.std(X_train_C1, axis=0, ddof=1)
    C2_std = np.std(X_train_C2, axis=0, ddof=1)
    C1_prob = len(y_train_C1) / len(y_train)
    C2_prob = len(y_train_C2) / len(y_train)
    for i in range(len(X_test)):
        g1 = ln(C1_prob) - 0.5 * np.sum(( (X_test[i] - C1_mean) / C1_std )**2)
        g2 = ln(C2_prob) - 0.5 * np.sum(( (X_test[i] - C2_mean) / C2_std )**2)
        if g1 > g2:
            y_pred[i] = 1
        else:
            y_pred[i] = 0
    return y_pred

y_pred2 = Gaussian_NB(X_train, y_train, X_test)
```

Separate C1(Survived) and C2(Dead).

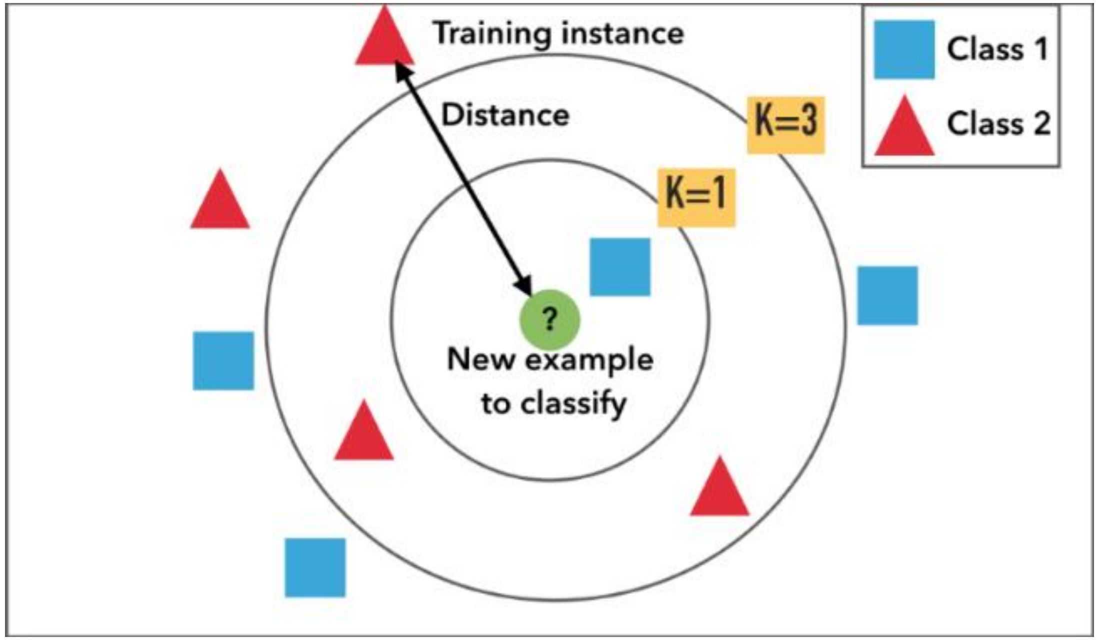
Calculate discriminant function and make prediction.

$$g_i(\mathbf{x}) = -\frac{1}{2}\sum_{j=1}^{d}\left(\frac{x_j^t - m_{ij}}{s_j}\right)^2 + \log \hat{P}(C_i)$$

Selection of Models – KNN

- K-Nearest Neighbor Classifier
  - Algorithm
    - A model that classifies data points based on the points that are most similar to it
    - Step 1: Calculate Euclidean Distance
    - Step 2: Get Nearest Neighbors (number of nearest neighbors, K=1,3,5,7,…)
    - Step 3: Make Predictions

$$Euclidean\ Disctance = \sqrt{\sum_{i}^{N}(x1_i - x2_i)^2}$$



Selection of Models – KNN

- Code from scratch

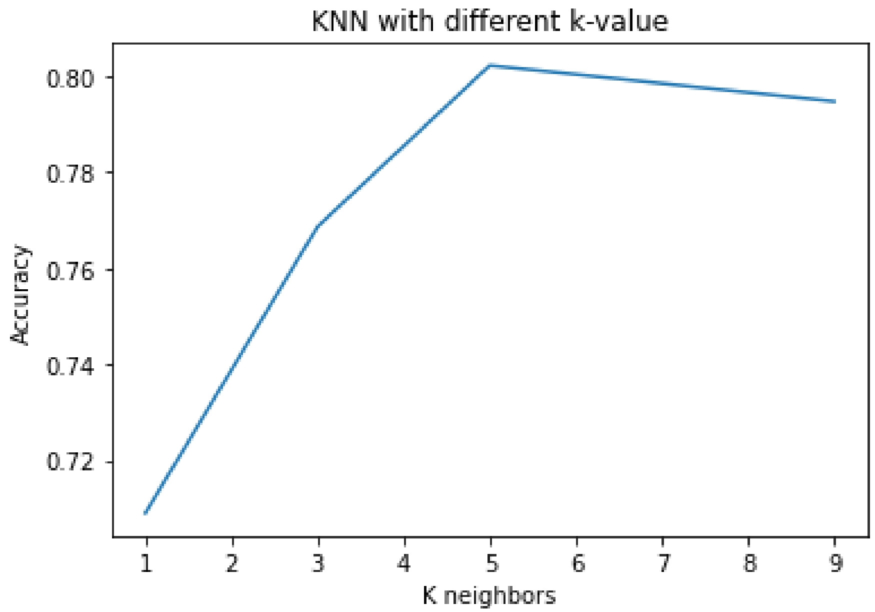
```python
#%% KNN
K_neighbors = 5
X_train = ( X_train - np.mean(X_train, axis=0) ) / np.std(X_train, axis=0, ddof=1)
X_test = ( X_test - np.mean(X_test, axis=0) ) / np.std(X_test, axis=0, ddof=1)

def KNN(X_train, y_train, X_test, k):
    y_pred = np.zeros(len(X_test))
    distances = pd.DataFrame(np.zeros([len(X_train),2]), columns=['y','dist'])
    distances['y'] = y_train
    for i in range(len(X_test)):
        for j in range(len(X_train)):
            distances['dist'][j] = np.sqrt(np.sum((X_test[i]-X_train[j])**2))
        dist = distances.sort_values(by=['dist'])[:k]
        y_pred[i] = dist['y'].value_counts().idxmax()
    return y_pred

y_pred1 = KNN(X_train, y_train, X_test, k=K_neighbors)
```

Normalization

- Our KNN model has the best performance when K=5, hence we set K=5.



Results & Discussion

	A	B	C	D	E	F
Pclass	**observed**	**observed**	**Observed**	**Observed**	**Depend on Fare**	**Depend on Fare**
Name	**A->1, B->2 …**	**A->1, B->2 …**	--	--	--	--
Age	**Mean age**	**Random number (mean +std) Age group(1-6)**	**Random number (mean +std) Age group(1-6)**	**Random number (mean +std) Age group(1-6)**	**Random number (mean +std) Age group(1-6)**	**Random number (mean +std) Age group(1-6)**
SibSp	+	--	--	+	+	+
Parch	+	--	--	+	+	+
relatives	--	+	+	+	+	--
Fare	+	+	**Fare band (4)**	**Fare band (4)**	**Fare per person**	**Fare per person**
Not alone	--	+	+	+	+	--
KNN (k=3)	0.79640	0.76047	0.76646	0.76047	0.75449	0.73053
KNN (k=5)	0.80239	0.77844	0.76646	0.76646	0.73056	0.75449
Naïve Bayes	0.81437	**0.82634**	0.77844	0.74850	0.77245	0.76047

Discussion

	A	B	C	D	E	F
KNN (k=3)	0.79640	0.76047	0.76646	0.76047	0.75449	0.73053
KNN (k=5)	0.80239	0.77844	0.76646	0.76646	0.73056	0.75449
Naïve Bayes	0.81437	**0.82634**	0.77844	0.74850	0.77245	0.76047

- **Gaussian Naïve Bayes Classifier v.s KNN**
  - **All the results of Naïve are better than KNN except "D"**

Gaussian Naïve Bayes considered every feature

$$\frac{P(x_1|y)P(x_2|y)P(x_3|y)\ \ldots\ P(x_n|y)\ P(y)}{P(x_1)P(x_2)P(x_3)\ \ldots\ P(x_n)}$$

Discussion

	A	B (B')	C
Pclass	observed	observed	Observed
Name	A->1, B->2 …	A->1, B->2 …(--)	--
Age	Mean age	Random number (mean +std) Age group(1-6)	Random number (mean +std) Age group(1-6)
SibSp	+	--	--
Parch	+	--	--
relatives	--	+	+
Fare	+	+	Fare band (4)
Not alone	--	+	+
Naïve Bayes	0.81437	**0.82634**	0.77844

- **Name**
  - **B' : delete the feature of Name**
  - **Result B and Result B' are the same**
- **Fare**
  - **Original fare value is more informative than fare band.**
- **SibSp and Parch**
  - **Number of relatives and Not alone have enough information.**

Discussion

Features	B0	B 1	B2	B 3	B 4	B5
Pclass	observed	observed	observed	observed	observed	Depend on Fare
Name	A->1, B->2 …	--	--	A->1, B->2 …	A->1, B->2 …	A->1, B->2 …
Age	Age group(1-6)	Age group(1-6)	Mean Age	Age group(1-6)	Age group(1-6)	Age group(1-6)
SibSp	+	+	+	+	+	+
Parch	+	+	+	+	+	+
Fare	+	+	+	Fare Band (4)	Fare per person	+
relatives	+	+	+	+	+	+
Not alone	+	+	+	+	+	+
Naïve Bayes	0.83	0.83	0.81	0.79	0.80	0.83

- "Name" did not affect the results. (B0,B1)
- Label of Pclass differently did not affect the results. (B0,B5)
- "Fare" need not label. (B3,B4)
- "Age" label to different group have better result. (B1,B2)

standard deviation (stdv): Fare > Age
"Fare" more discrete
→ need not divided into groups

Conclusion

- Feature selection and feature labeling are the most important step for machine learning.
    - Trial and error

- Different algorithms and models have different benefit.
    - Gaussian NB : Fast, Easy to train.
    - KNN : Simple, Intuitive.

References

https://towardsdatascience.com/introduction-to-na%C3%AFve-bayes-classifier-fa59e3e24aaf

https://www.kaggle.com/nadintamer/titanic-survival-predictions-beginner

https://www.youtube.com/watch?v=UqYde-LULfs

https://jingwen-z.github.io/titanic-survival-prediction/

https://medium.com/@lope.ai/knn-classifier-from-scratch-with-numpy-python-5c436e26a228

https://www.geeksforgeeks.org/naive-bayes-classifiers/