Department of Computer Engineering

CSE 454 – Data Mining

Fall 2021 – 2022

Final Project Report

21.01.2022

Esra Eryılmaz

171044046

Demonstration Video Link

https://youtu.be/vbrL36jljN4

Note : I write all my implementations on the jupyter-notebook environment and I made an effort to put comments on it.

Problem Definition

The problem is create a model that predicts which passengers survived the Titanic shipwreck with using data mining algorithms.

I implement Logistic Regression classification algorithm on Titanic dataset and evaluate performance with other models using the sklearn library.

- **Dataset :** Titanic Dataset

The data has been split into two groups:

- training set (train.csv)
- test set (test.csv)

The training set used to build my model.

The test set used to see how well my model performs on unseen data. For each passenger in the test set, I used the model I trained to predict whether or not they survived the sinking of the Titanic.

Analyze the dataset

- Preview the dataset

```
# preview the data
train.head()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

```
train.tail()
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

- There are 891 columns and 12 attributes in the dataset.

```
print(train.columns.values)
```

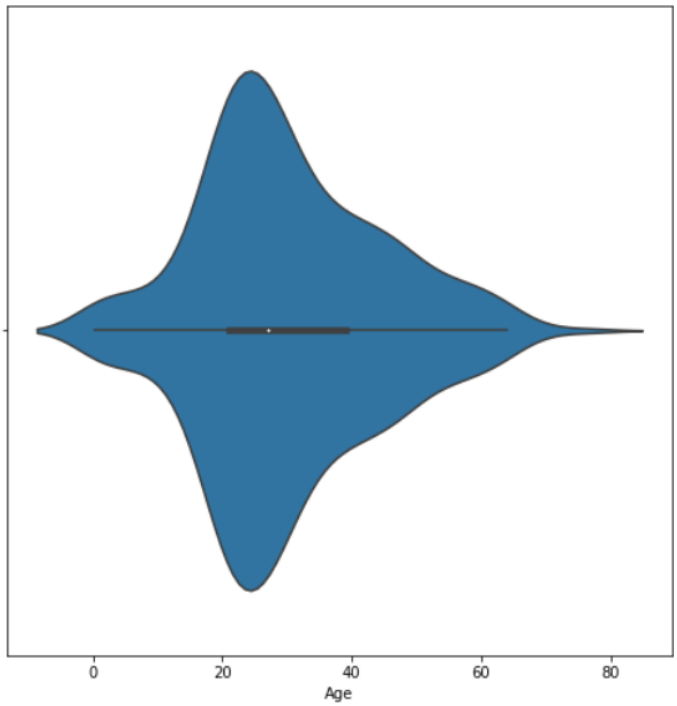
```
['PassengerId' 'Survived' 'Pclass' 'Name' 'Sex' 'Age' 'SibSp' 'Parch'
 'Ticket' 'Fare' 'Cabin' 'Embarked']
```

- Three attributes (Age, Cabin, Embarked) has null values and I did preprocessing on that attirubutes.

```
train.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

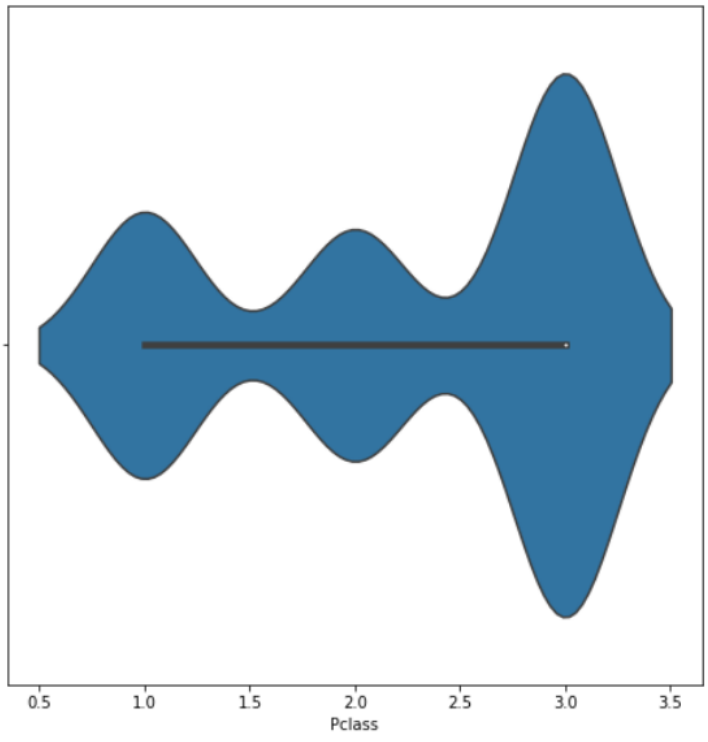
- **Passengers by age :** There are many passengers between the ages of 20 and 30



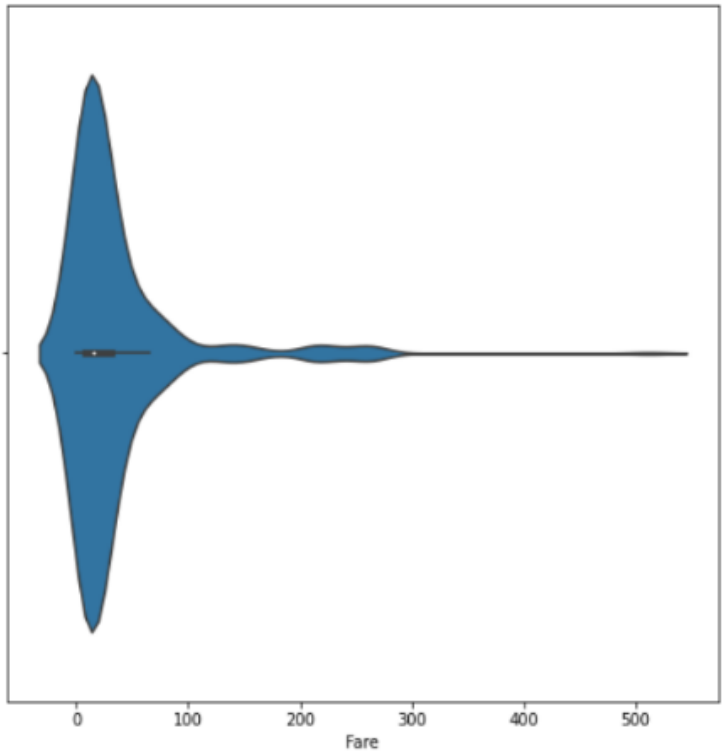
- **Passengers by Pclass :** A proxy for socio-economic status

1st = Upper
2nd = Middle
3rd = Lower

Third class passenger number is high.



- **Passengers by Fare :** Passenger fare (pound)
  There are more passengers paying low fares because there are more 3rd class passengers.

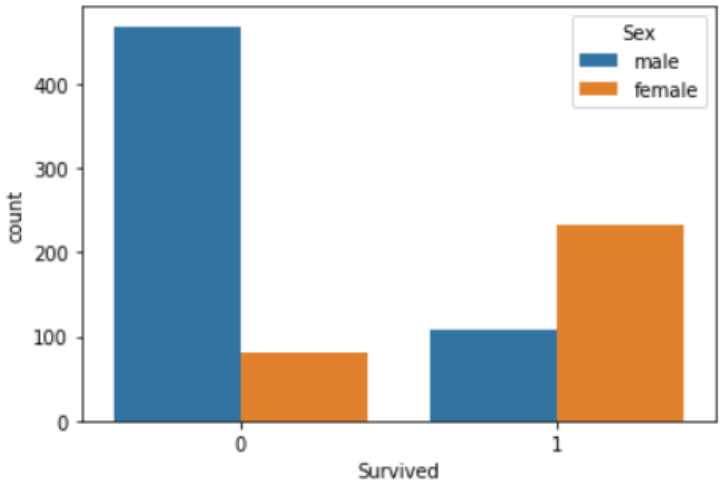


- **Correlation Matrix :** The strongest correlations to survival are Pclass (negative correlation), Sex (positive correlation), and Fare (positive correlation). The higher the fare, the higher the survival. Pclass is negatively correlated because in this context a smaller Pclass denotes a higher level of service class.

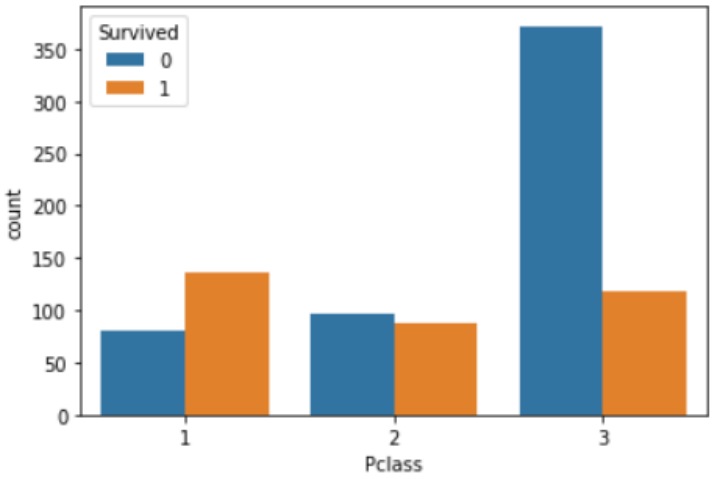
```
# Checking the Correlation Matrix
corr = train.corr()
corr.style.background_gradient(cmap='Greys').set_precision(2)
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
PassengerId	1.00	-0.01	-0.04	0.04	-0.06	-0.00	0.01
Survived	-0.01	1.00	-0.34	-0.08	-0.04	0.08	0.26
Pclass	-0.04	-0.34	1.00	-0.37	0.08	0.02	-0.55
Age	0.04	-0.08	-0.37	1.00	-0.31	-0.19	0.10
SibSp	-0.06	-0.04	0.08	-0.31	1.00	0.41	0.16
Parch	-0.00	0.08	0.02	-0.19	0.41	1.00	0.22
Fare	0.01	0.26	-0.55	0.10	0.16	0.22	1.00

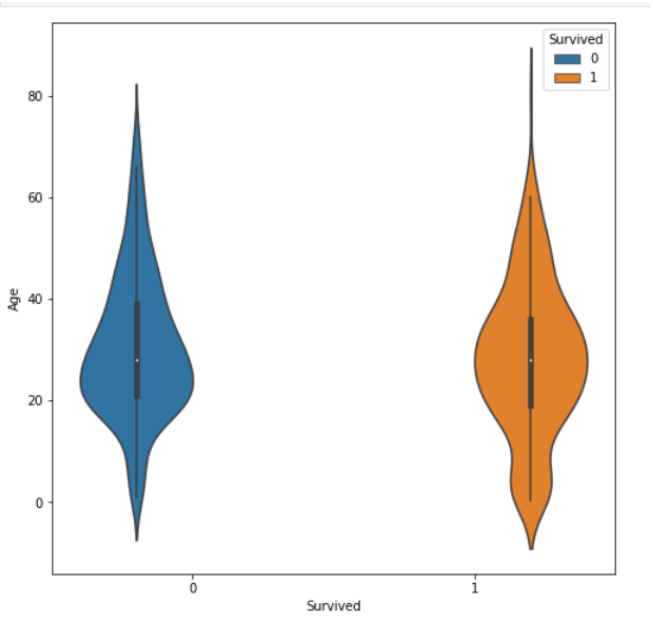
- Checking how many people survived according to sex



- Checking how many people survived according to Pclass



- Checking how many people survived according to Age



Data Preprocessing

It take considerable amount of processing time. Examples of data preprocessing include **cleaning, instance selection, normalization, transformation, feature extraction and selection, etc**. The product of data preprocessing is the final training set.

- ## Data cleaning for missing data

- **Age** has some missing values. I replace Age null values with median value. While putting the median value, I split Age values into 3 different median category according to the Passenger classes. Because for example first class average age is higher than the third class passengers.

- **Cabin** has 687 missing values (75% of dataset). I drop this column altogether.

```
train.drop('Cabin', axis=1, inplace=True)
test.drop('Cabin', axis=1, inplace=True)
```

- **Embarked** has some missing values. Embarked has three different categories. C = Cherbourg, Q = Queenstown, S = Southampton. The most of the dataset has 'S' value for Embarked fort hat reason I replace Embarked null values with 'S'.

```
train['Embarked'] = train['Embarked'].fillna('S')
```

- **Fare** has just one missing value on test set. I replace null with median.

- ## Feature Selection

- SibSp = # of siblings / spouses aboard the Titanic

Parch = # of parents / children aboard the Titanic

SibSp and Parch can be combined, I create a new "FamilySize" column.

- **Ticket** column likely doesn't have any correlation to survival. If there is, that relationship is already captured by Fare and PClass. I drop Ticket column.

- Drop **Name** column, with the same reason as Ticket.

```
# creating FamilySize column
train['FamilySize'] = train['SibSp'] + train['Parch'] + 1
test['FamilySize'] = test['SibSp'] + test['Parch'] + 1

columns = ['SibSp', 'Parch', 'Ticket', 'Name']
train.drop(columns, axis=1, inplace=True)
test.drop(columns, axis=1, inplace=True)
```

- I used one-hot encoding to make "**Embarked**" values into categorical values. {'C', 'Q', 'S'}

I used one-hot encoding to make "**Sex**" values into categorical values. {"male" : 0, "female" : 1}

(pandas.get_dummies() is used for data manipulation. It converts categorical data into dummy)

- **Feature Normalization**

- Data Normalization (MinMaxScaler() function does)

```
scaler = preprocessing.MinMaxScaler()
```

- Split x_train, x_test, y_train, y_test

(from sklearn.model_selection import train_test_split)

```
# standardizing x_train and test set
x_to_scale = scaler.fit_transform(x_to_scale)
x = np.concatenate((x_to_scale, x_categories), axis = 1)

test_to_scale = scaler.fit_transform(test_to_scale)
test = np.concatenate((test_to_scale, test_categories), axis = 1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 40)
```

Data analyze and data preprocessing done !

Logistic Regression Classification implementation

- What is logistic regression ?

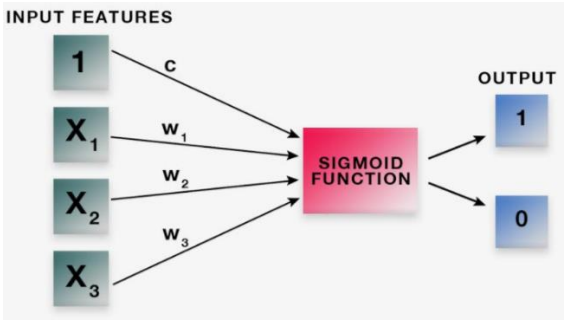
A logistic regression model is a classification model that predicts the probability of an outcome or an event. It does so by fitting the data (independent variables) to the logistic function. As an example, logistic regression can be used to predict the probability of a patient having heart disease, given the patient's age, sex, cholesterol level, blood pressure, and so on.

The logistic function is defined by a sigmoid function, which takes any real number input and output value between 0 and 1. Large positive values will output 1, and negative numbers will output 0.

Logistic regression includes following functions:

(These functions mostly consists of a mathematical expressions)

- **Sigmoid function** = Logistic Regression function



$$\phi(z) = \frac{1}{1+e^{-z}}$$

- **Cost function**

$$J(\boldsymbol{\theta}) = \frac{1}{m}\sum_{i=1}^{m} \text{Cost}(h_\theta(x), y)$$

$$\text{Cost}(h_\theta(\mathbf{x}), y) = -y log(h_\theta(\mathbf{x})) - (1-y) log(1-h_\theta(\mathbf{x}))$$

- **Gradient descent function**. This function returns the best parameters that minimizes our cost function. To minimize my cost, I used Gradient Descent just like in Linear Regression implementation.

Gradient descent for logistic regression:

while not converged {

$$\theta_j^{\text{new}} = \theta_j^{\text{old}} - \alpha \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \text{ for } j = 0, 1, \ldots, n$$

}

- **Predict function** for predicting Survival. When our sigmoid function outputs a value more than or equal to 0.5, it will output 1 for "Survive", and if the value is less than 0.5, it will output "0" for did not survive.

Theta : Three function takes theta for parameter. I used scipy.optimize library for finding optimum parameters with fmin_bfgs()

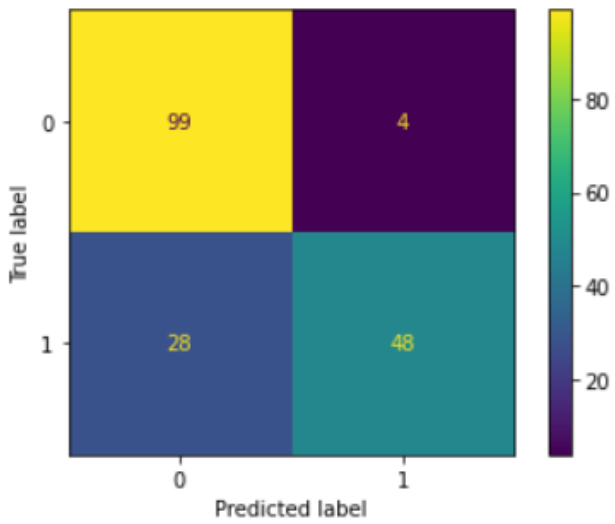
```
#initializing theta
initial_theta = np.random.rand(x_train.shape[1], 1)

#finding optimum parameters with fmin_bfgs
import scipy.optimize

optimized_theta = scipy.optimize.fmin_bfgs(costFunction, fpri
```

My implementation results :

	precision	recall	f1-score	support
0	0.82	0.94	0.88	446
1	0.87	0.65	0.74	266
accuracy			0.83	712
macro avg	0.85	0.80	0.81	712
weighted avg	0.84	0.83	0.83	712

```
F1 score for Logistic regression implementation :  0.7429805615550756
Accuracy for Logistic regression implementation :  0.8328651685393258
```

Confusion Matrix :



Other Models (from sklearn library)

I used library for these models and calculate accuracy and f1 score for each one.

- Logistic Regression
- KNN (K-Nearest Neighbors)
- SVM (Support Vector Machines)
- Naive Bayes classifier
- Decision Tree
- Random Forest
- Perceptron
- AdaBoost

Evaluate results

	Model	Score	F1 score
1	KNN	87.78	0.805369
5	Random Forest	98.17	0.805369
4	Decision Tree	98.31	0.763158
0	Logistic Regression	81.60	0.755556
2	SVM	80.90	0.750000
7	AdaBoost	88.06	0.750000
3	Naive Bayes	78.93	0.739726
6	Perceptron	73.88	0.518519

My model got an F1 score of 0.74.

My model got an Accuracy of 83%.

By looking at my implementation of logistic regression and sklearn library implementation, I think their results are similar.

- My F1 score 0.74 and library F1 score 0.75
- My accuracy 83% and library accuracy 81%

For that reason I don't think my imlementation has any error.

And with looking at other model results I see that for the F1 score the the best model is KNN and Random Forest. Both have 0.80 F1 score which is interesting.

For the Random Forest at first I want to implement that algorithm but I couldn't, it is hard to do it and can understand that algorithm give me good result because of difficulty and also it is a ensemble method too. But KNN is simple to implement and that algorithm results are generally good which is good too and also interesting.

And the worst model fort he Titanic dataset is Perceptron.

And also I see that by looking at results accuracy and F1 score not very similar. For example Decision Tree has best accuracy but that algorithm is 3rd place in the F1 score.

Academic Paper Summary

An Introduction to Logistic Regression Analysis and Reporting

CHAO-YING JOANNE PENG
KUK LIDA LEE
GARY M. INGERSOLL

Logistic regression can be a powerful analytical technique. Logistic regression was started to used in the late 1960s and early 1970s. And it became routinely available in statistical packages in the early 1980s.

In that time the use of logistic regression has increased in the social sciences. Generally, logistic regression is well suited for describing and testing hypotheses about relationships between a categorical outcome variable and one or more categorical or continuous predictor variables.

Logistic regression is an alternative method to use other than the simpler linear regression.

Although logistic regression can do categorical outcomes that are polytomous, In this article they focus on dichotomous outcomes only.

Logistic regression uses the concept of odds ratios to calculate the probability.

The simple logistic model has the form :

$$\text{logit}(Y) = \textit{natural}\log(\textit{odds}) = \ln\left(\frac{\pi}{1-\pi}\right) = \alpha + \beta X.$$

For calculating effectiveness of the model we should do : (a) overall model evaluation, (b) statistical tests of individual predictors, (c) goodness-of-fit statistics and (d) validations of predicted probabilities.

The result gives an 'S' shaped curve to model the data.

Paper Reference :

[1] Peng, Joanne & Lee, Kuk & Ingersoll, Gary. (2002). An Introduction to Logistic Regression Analysis and Reporting. Journal of Educational Research - J EDUC RES. 96. 3-14. 10.1080/00220670209598786.