Соревнование в Kaggle "Титаник"

https://www.kaggle.com/c/titanic/

Цель соревнования: исходя из данных пассажира, таких как его пол, возраст, номер билета и так далее, предсказать, выживет он при крушении Титаника или нет. Задача являет собой простейшую задачу бинарной классификации, то есть мы классифицируем пассажиров либо как выживших, либо как погибших. Данные, представленные в соревновании, разбиты на две группы:
1) Обучающая выборка (train.csv)
2) Тестовая выборка (test.csv)
Обучающая выборка используется, как понятно из названия, для обучения модели, для тестовой нужно получить результаты, чтобы проверить, как модель работает с теми данными, которые не участвовали в обучении.

1.Подготовка данных

In [199]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df_train = pd.read_csv('titanic/train.csv',index_col=0)
df_test = pd.read_csv('titanic/test.csv',index_col=0)
df_train
```

Out[199]:

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
...	...	...	...	...	...	...	...	...	...	...	...
887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.0000	NaN	S
888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.0000	B42	S
889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.4500	NaN	S
890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.0000	C148	C
891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.7500	NaN	Q

891 rows × 11 columns

Остановимся на данных обучающей выборки подробнее. Здесь параметр **"Survived"** отвечает за то, выжил человек или нет, то есть при обучении с учителем данный столбец являет собой **выходные данные**. Всё остальное является **входными данными.**
Для того, чтобы задачу решить, данные необходимо обработать. Если с выходными данными ясно - если человек выжил, то 1, если нет, то 0, то с входными несколько сложнее.

Например, столбец **"Name"** отвечает за идентификацию пассажиров. Так как имя в общем случае не влияет на возможную смерть человека, а все пассажиры уже пронумерованы, то этот столбец можно исключить. То же самое можно сказать и про столбец **"Ticket"**, он также отвечает лишь за идентификацию пассажира. Рассмотрим теперь столбец **"Cabin"**

In [200]:
```
print('Число неопределенных значений в столбце "Cabin"',df_train['Cabin'].isna().sum())
```

```
Число неопределенных значений в столбце "Cabin" 687
```

Мы видим, что для подавляющего большинства случаев это значение не определено. Если бы их количество было небольшим, можно было бы заменить недостающее значение на наиболее встречающиеся, но так как их большинство, этот столбец тоже придется отбросить, так как никакой информации он нам не дает.

Таким образом, несущие смысл столбцы для входных данных:

"Pclass" - Класс билета, где 1 = первый, 2 = второй, 3 = третий классы,

"Sex" - Пол пассажира,

"Age" - Возраст пассажира,

"SibSp" - Количество братьев и сестер или супругов на борту Титаника,

"Parch" - Количество родителей или детей на борту Титаника,

"Fare" - Пассажирский тариф,

"Embarked" - Порт погрузки на корабль, где C = Cherbourg, Q = Queenstown, S = Southampton.

Тогда:

In [201]:
```
X_train = df_train.drop(['Survived', 'Name', 'Ticket', 'Cabin'], axis=1)

X_test = df_test.drop([ 'Name', 'Ticket', 'Cabin'], axis=1)

Y_train = df_train['Survived']

X_train
```

Out[201]:

PassengerId	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
1	3	male	22.0	1	0	7.2500	S
2	1	female	38.0	1	0	71.2833	C
3	3	female	26.0	0	0	7.9250	S
4	1	female	35.0	1	0	53.1000	S
5	3	male	35.0	0	0	8.0500	S
...	...	...	...	...	...	...	...
887	2	male	27.0	0	0	13.0000	S
888	1	female	19.0	0	0	30.0000	S
889	3	female	NaN	1	2	23.4500	S
890	1	male	26.0	0	0	30.0000	C
891	3	male	32.0	0	0	7.7500	Q

891 rows × 7 columns

In [202]:
```
Y_train
```

Out[202]:
```
PassengerId
1      0
2      1
3      1
4      1
5      0
      ..
887    0
888    1
889    0
890    1
891    0
Name: Survived, Length: 891, dtype: int64
```

Для того, чтобы работа с данными была корректна, необходимо проверить, все ли данные присутствуют в обучающей и тестовой выборках:

In [203]:
```
print("Обучающая выборка")
cols= X_train.columns
for i in cols:
    print('Неопределенные значения в "',i,'":',X_train[i].isna().sum())
print()
print("Тестовая выборка")
for i in cols:
    print('Неопределенные значения в "',i,'":',X_test[i].isna().sum())
print()
print("Неопределенные значения в выходных данных тестовой выборки:",Y_train.isna().sum())
```

```
Обучающая выборка
Неопределенные значения в " Pclass ": 0
Неопределенные значения в " Sex ": 0
Неопределенные значения в " Age ": 177
Неопределенные значения в " SibSp ": 0
Неопределенные значения в " Parch ": 0
Неопределенные значения в " Fare ": 0
Неопределенные значения в " Embarked ": 2

Тестовая выборка
Неопределенные значения в " Pclass ": 0
Неопределенные значения в " Sex ": 0
Неопределенные значения в " Age ": 86
Неопределенные значения в " SibSp ": 0
Неопределенные значения в " Parch ": 0
Неопределенные значения в " Fare ": 1
Неопределенные значения в " Embarked ": 0

Неопределенные значения в выходных данных тестовой выборки: 0
```

Недостающие данные нужно заполнить медианными значениями по столбцам. Однако сначала текстовые данные необходимо перевести в числовые, так как наши модели будут работать именно с числовыми данными:

In [204]:
```
def prepare_dataframe(df):
    df_tmp = df.drop(['Sex', 'Embarked', 'Pclass'], axis=1)
    df_1 = pd.get_dummies(df['Sex'])
    df_2 = pd.get_dummies(df['Embarked'], prefix='Embarked')
    df_3 = pd.get_dummies(df['Pclass'], prefix='Pclass')
    df_tmp = pd.concat((df_tmp, df_1, df_2, df_3), axis=1)
    return df_tmp
```

In [205]:
```
X_prepared_train=prepare_dataframe(X_train)
X_prepared_test=prepare_dataframe(X_test)
X_prepared_train
```

Out[205]:

PassengerId	Age	SibSp	Parch	Fare	female	male	Embarked_C	Embarked_Q	Embarked_S	Pclass_1	Pclass_2	Pclass_3
1	22.0	1	0	7.2500	0	1	0	0	1	0	0	1
2	38.0	1	0	71.2833	1	0	1	0	0	1	0	0
3	26.0	0	0	7.9250	1	0	0	0	1	0	0	1
4	35.0	1	0	53.1000	1	0	0	0	1	1	0	0
5	35.0	0	0	8.0500	0	1	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
887	27.0	0	0	13.0000	0	1	0	0	1	0	1	0
888	19.0	0	0	30.0000	1	0	0	0	1	1	0	0
889	NaN	1	2	23.4500	1	0	0	0	1	0	0	1
890	26.0	0	0	30.0000	0	1	1	0	0	1	0	0
891	32.0	0	0	7.7500	0	1	0	1	0	0	0	1

891 rows × 12 columns

Для каждого категориального признака был создан свой столбец, содержащий бинарные данные. Теперь заполним медианными значениями:

In [206]:
```
X_prepared_train = X_prepared_train.fillna(X_prepared_train.median())
X_prepared_test = X_prepared_test.fillna(X_prepared_test.median())
print("Обучающая выборка")
cols= X_prepared_train.columns
for i in cols:
    print('Неопределенные значения в "',i,'":',X_prepared_train[i].isna().sum())
print()
X_prepared_train
```

```
Обучающая выборка
Неопределенные значения в " Age ": 0
Неопределенные значения в " SibSp ": 0
Неопределенные значения в " Parch ": 0
Неопределенные значения в " Fare ": 0
Неопределенные значения в " female ": 0
Неопределенные значения в " male ": 0
Неопределенные значения в " Embarked_C ": 0
Неопределенные значения в " Embarked_Q ": 0
Неопределенные значения в " Embarked_S ": 0
Неопределенные значения в " Pclass_1 ": 0
Неопределенные значения в " Pclass_2 ": 0
Неопределенные значения в " Pclass_3 ": 0
```

Out[206]:

PassengerId	Age	SibSp	Parch	Fare	female	male	Embarked_C	Embarked_Q	Embarked_S	Pclass_1	Pclass_2	Pclass_3
1	22.0	1	0	7.2500	0	1	0	0	1	0	0	1
2	38.0	1	0	71.2833	1	0	1	0	0	1	0	0
3	26.0	0	0	7.9250	1	0	0	0	1	0	0	1
4	35.0	1	0	53.1000	1	0	0	0	1	1	0	0
5	35.0	0	0	8.0500	0	1	0	0	1	0	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
887	27.0	0	0	13.0000	0	1	0	0	1	0	1	0
888	19.0	0	0	30.0000	1	0	0	0	1	1	0	0
889	28.0	1	2	23.4500	1	0	0	0	1	0	0	1
890	26.0	0	0	30.0000	0	1	1	0	0	1	0	0
891	32.0	0	0	7.7500	0	1	0	1	0	0	0	1

891 rows × 12 columns

Также на всякий случай нормализуем наши данные. Некоторые модели могут работать с ненормализованными данными, но некоторые - нет.

In [208]:
```
from sklearn.preprocessing import MinMaxScaler
scalerX = MinMaxScaler()
X_scaled_train = scalerX.fit_transform(X_prepared_train)
X_scaled_test = scalerX.transform(X_prepared_test)
pd.DataFrame(X_scaled_train,columns=X_prepared_train.columns)
```

Out[208]:

	Age	SibSp	Parch	Fare	female	male	Embarked_C	Embarked_Q	Embarked_S	Pclass_1	Pclass_2	Pclass_3
0	0.271174	0.125	0.000000	0.014151	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
1	0.472229	0.125	0.000000	0.139136	1.0	0.0	1.0	0.0	0.0	1.0	0.0	0.0
2	0.321438	0.000	0.000000	0.015469	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
3	0.434531	0.125	0.000000	0.103644	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
4	0.434531	0.000	0.000000	0.015713	0.0	1.0	0.0	0.0	1.0	0.0	0.0	1.0
...	...	...	...	...	...	...	...	...	...	...	...	...
886	0.334004	0.000	0.000000	0.025374	0.0	1.0	0.0	0.0	1.0	0.0	1.0	0.0
887	0.233476	0.000	0.000000	0.058556	1.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0
888	0.346569	0.125	0.333333	0.045771	1.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0
889	0.321438	0.000	0.000000	0.058556	0.0	1.0	1.0	0.0	0.0	1.0	0.0	0.0
890	0.396833	0.000	0.000000	0.015127	0.0	1.0	0.0	1.0	0.0	0.0	0.0	1.0

891 rows × 12 columns

В таблице выше показано, что данные были нормализованы от 0 до 1. Теперь входные и выходные данные полностью обработаны.

2.Статистический анализ данных

В таблице выше показано, что данные были нормализованы от 0 до 1. Теперь входные и выходные данные полностью обработаны.

Однако, прежде чем приступать к машинному обучению, попробуем проанализировать исходные данные сами и сделать на основе анализа предположения касательно выживаемости пассажиров. Для начала, попробуем исследовать, как на выживаемость пассажира влияет класс его билета:

In [209]:
```
X_a= X_train.copy()
X_a['Survived']=Y_train
```

In [210]:
```
X_a[['Pclass', 'Survived']].groupby(['Pclass'], as_index=False).mean()
```

Out[210]:

	Pclass	Survived
0	1	0.629630
1	2	0.472826
2	3	0.242363

Мы увидели, что у пассажиры из первого класса имеют шанс выжить ~63%, пассажиры из второго класса меньше, ~47%, а пассажиры третьего класса выживут с вероятность всего лишь 24%. Теперь посмотрим, как на выживаемость влияет пол пассажиров:

In [211]:
```
X_a[['Sex', 'Survived']].groupby(['Sex'], as_index=False).mean()
```

Out[211]:

	Sex	Survived
0	female	0.742038
1	male	0.188908

Здесь мы видим, что на титанике плавали в основном джентльмены, ибо вероятность выжить у женщин составляет ~74%, в то время как у мужчин ~19%. Теперь проанализируем влияние возраста:

In [212]:
```
print("Шанс выжить у младенца до года - ",round(
      X_a[X_a['Age']<1]['Survived'].mean()
                                       *100,1),"%")
print("Шанс выжить у детей от года до 18 лет - ",round(
      X_a[(X_a['Age']>1)&(X_a['Age']<18)]['Survived'].mean()
                                       *100,1),"%")
print("Шанс выжить у взрослых от 18 до 60 лет - ",round(
      X_a[(X_a['Age']>18)&(X_a['Age']<60)]['Survived'].mean()
                                       *100,1),"%")
print("Шанс выжить у пожилых людей старше 60 лет - ",round(
      X_a[X_a['Age']>60]['Survived'].mean()
                                       *100,1),"%")
```

```
Шанс выжить у младенца до года -  100.0 %
Шанс выжить у детей от года до 18 лет -  49.5 %
Шанс выжить у взрослых от 18 до 60 лет -  38.8 %
Шанс выжить у пожилых людей старше 60 лет -  22.7 %
```

Видим, что приоритетом при спасении являются младенцы и дети до 18 лет, а чем старше человек - тем меньше шансов у него спастись. Теперь посмотрим, как влияет наличие супругов или братьев и сестер на выживаемость.

In [213]:
```
X_a[['SibSp', 'Survived']].groupby(['SibSp'], as_index=False).mean()
```

Out[213]:

	SibSp	Survived
0	0	0.345395
1	1	0.535885
2	2	0.464286
3	3	0.250000
4	4	0.166667
5	5	0.000000
6	8	0.000000

Видим, что чем наличие одного близкого человека рядом увеличивает вероятность выживания до ~53.5%. Теперь исследуем влияние количества родителей или детей пассажира, находящихся на борту:

In [214]:
```
X_a[['Parch', 'Survived']].groupby(['Parch'], as_index=False).mean()
```

Out[214]:

	Parch	Survived
0	0	0.343658
1	1	0.550847
2	2	0.500000
3	3	0.600000
4	4	0.000000
5	5	0.200000
6	6	0.000000

Видим, что наибольшие шансы выжить имеют люди, у которых количество данных родственников достигает трёх и составляют они 60%. Теперь посмотрим, как влияет тариф пассажира.

In [215]:
```
print("Шанс выжить у пассажира с тарифом ниже среднего - ",round(
      X_a[X_a['Fare']<X_a['Fare'].mean()]['Survived'].mean()
                                       *100,1),"%")
print("Шанс выжить у пассажира с тарифом выше среднего - ",round(
      X_a[X_a['Fare']>X_a['Fare'].mean()]['Survived'].mean()
                                       *100,1),"%")
```

```
Шанс выжить у пассажира с тарифом ниже среднего -  31.8 %
Шанс выжить у пассажира с тарифом выше среднего -  59.7 %
```

Итого, чем дороже билет, тем больше шанс выжить. И, наконец, выясним, как влияет порт посадки на выживаемость.

In [216]:
```
X_a[['Embarked', 'Survived']].groupby(['Embarked'], as_index=False).mean()
```

Out[216]:

	Embarked	Survived
0	C	0.553571
1	Q	0.389610
2	S	0.336957

Итого, видим, что если пассажир сел на корабль в порту С, то его шансы сильно выше, чем в других портах. Теперь выясним, какова будет вероятность выжить у девушки до 18 лет:

In [311]:
```
print(round(X_a[(X_a['Age']<18)&(X_a['Sex']=='female')]['Survived'].mean()*100,1),"%")
```

```
69.1 %
```

Теперь посмотрим, какова вероятность выжить у мужчины сташе 18:

In [312]:
```
print(round(X_a[(X_a['Age']>18)&(X_a['Sex']=='male')]['Survived'].mean()*100,1),"%")
```

```
18.1 %
```

Итого, при помощи статистического анализа мы сделали два предположения по поводу выживаемости пассажиров, которые и будем проверять при помощи машинного обучения. В силу небольшой выборки были сделаны более общие предположения, но при достаточном количестве данных их можно было бы сделать более точными.

3.Машинное обучение

Обучать будем несколько моделей классификации, среди которых будет несколько деревьев решений (XGBoost, Random Forest),...
Так как все данные подготовлены в пункте 1, можем начинать обучение.

In [324]:
```
def round1(r2):
    return round(r2,3)
```

In [332]:
```
hypothesis_1=scalerX.transform(X_prepared_train[(X_prepared_train['Age']<18)&(X_prepared_train['female'
]==1)])
hypothesis_2=scalerX.transform(X_prepared_train[(X_prepared_train['Age']>18)&(X_prepared_train['male']=
=1)])
```

Линейная регрессия:

In [333]:
```
from sklearn.linear_model import LinearRegression

linear_regression = LinearRegression()
linear_regression.fit(X_scaled_train, Y_train)
pred_lr=linear_regression.predict(X_scaled_train)
print("Гипотеза 1:", 100*round1((linear_regression.predict(hypothesis_1)).mean()),"%" )
print("Гипотеза 2:", 100*round1((linear_regression.predict(hypothesis_2)).mean()),"%" )
```

```
Гипотеза 1: 75.7 %
Гипотеза 2: 19.400000000000002 %
```

Логистическая регрессия:

In [334]:
```
logistic_regression = LogisticRegression()
logistic_regression.fit(X_scaled_train, Y_train)
pred_lf=logistic_regression.predict(X_scaled_train)
print("Гипотеза 1:", 100*round1((logistic_regression.predict(hypothesis_1)).mean()),"%" )
print("Гипотеза 2:", 100*round1((logistic_regression.predict(hypothesis_2)).mean()),"%" )
```

```
Гипотеза 1: 92.7 %
Гипотеза 2: 4.3 %
```

Случайный лес(дерево решений):

In [335]:
```
from sklearn.ensemble import RandomForestClassifier
random_forest = RandomForestClassifier(n_estimators = 100)
random_forest.fit(X_scaled_train, Y_train)
print("Гипотеза 1:", 100*round1((random_forest.predict(hypothesis_1)).mean()),"%" )
print("Гипотеза 2:", 100*round1((random_forest.predict(hypothesis_2)).mean()),"%" )
```

```
Гипотеза 1: 69.1 %
Гипотеза 2: 15.4 %
```

XGBoost (дерево решений с использованием метода градиентного бустинга):

In [336]:
```
from xgboost import XGBClassifier
import warnings
warnings.filterwarnings('ignore')

XGB = XGBClassifier(n_estimators=100, max_depth=6, eta=0.3, gamma=0, learning_rate=0.3)
XGB.fit(X_scaled_train, Y_train)
pred_xgb=XGB.predict(X_scaled_train)
print("Гипотеза 1:", 100*round1((XGB.predict(hypothesis_1)).mean()),"%" )
print("Гипотеза 2:", 100*round1((XGB.predict(hypothesis_2)).mean()),"%" )
```

```
Гипотеза 1: 69.1 %
Гипотеза 2: 14.2 %
```

Итого, мы построили 4 модели и попробовали подтвердить две наши гипотезы, сделанные в пункте 2(выживание девушки младше 18 лет и мужчины старше 18 лет).Теперь необходимо понять, какая из моделей предсказывает результат лучше.Для этого предскажем на обучающей выборкой и сравним результат с известными значениями при помощи R2.

In [337]:
```
from sklearn.metrics import r2_score



print("Линейная регрессия:",round1(r2_score(Y_train,pred_lr)))
print("Логистическая регрессия:",round1(r2_score(Y_train,pred_lf)))
print("Случайный лес:",round1(r2_score(Y_train,pred_rf)))
print("XGB:",round1(r2_score(Y_train,pred_xgb)))
```

```
Линейная регрессия: 0.398
Логистическая регрессия: 0.155
Случайный лес: 0.915
XGB: 0.848
```

Как мы видим, самой лучшей моделью оказался случайный лес (Random Forest). Сравним результат наших гипотез и его результат их предсказания: **Предположения:**

Гипотеза 1: 69.1 %

Гипотеза 2: 18.1 %

RF:

Гипотеза 1: 69.1 %

Гипотеза 2: 15.4%

Теперь проверим те же гипотезы на тестовой выборке:

In [339]:
```
hypothesis_1_test=scalerX.transform(X_prepared_test[(X_prepared_test['Age']<18)&(X_prepared_test['femal
e']==1)])
hypothesis_2_test=scalerX.transform(X_prepared_test[(X_prepared_test['Age']>18)&(X_prepared_test['male'
]==1)])
print("Гипотеза 1 на тестовой выборке:", 100*round1((random_forest.predict(hypothesis_1_test)).mean()),
"%" )
print("Гипотеза 2 на тестовой выборке:", 100*round1((random_forest.predict(hypothesis_2_test)).mean()),
"%" )
```

```
Гипотеза 1 на тестовой выборке: 88.2 %
Гипотеза 2 на тестовой выборке: 14.000000000000002 %
```

Соответственно,аналогично находятся решения для других условий, гипотез, или просто для всей тестовой выборки

Как мы видим, те оценки, которые мы получили при помощи статистического анализа, близки к тем, которые даёт машинное обучение на обучающей выборке. Таким образом, мы проверили воспроизводимость результатов двумя способами и смогли получить интересующее нас решение на тестовой выборке.

In [ ]:
```

```