1. 查看特征信息

train\_df.info() train\_df.head()

–> 哪些特征是 categorical/numerical?

-> 哪些特征有 null values? -> 需要填充

1. 数据归类/可视化 -> 分析哪些特征具有重要影响，作出假设
2. 各特征分类下数据个数

train\_df['Pclass'].value\_counts()

train.groupby('passenger\_count').size()

sns.countplot(train\_df['Pclass'], hue=train['Survived'])

1. 两个特征数据分布

pd.crosstab(train\_df['Title'],train\_df['Sex'])

1. 单个特征对结果的影响

train\_df[['Pclass','Survived']].groupby('Pclass').mean()

连续变量 -> 分段观察pd.cut or pd.qcut

pd.qcut(train\_df['Age'],5).value\_counts()

train\_df['Survived'].groupby(pd.qcut(train\_df['Age'],5)).mean()

1. 是否为null对结果影响

train\_df[‘Survived’].groupby(train\_df[‘Cabin’].isnull()).mean()

1. 数据清洗
2. 查看数据范围，删除异常值
3. 补充空缺的数据：抽取类似数据的中位数/众数

group = combine\_df.groupby(['Title', 'Pclass'])['Age']

combine\_df['Age'] = group.transform(lambda x: x.fillna(x.median()))

1. 将连续变量分区间

bins = (0,6,12,18,25,35,60,100)

train\_df.Age = pd.cut(train\_df['Age'],bins)

1. 提取数据中有用的部分

combine\_df['Surname'] = combine\_df['Name'].apply(lambda x:x.split(',')[0])

combine\_df['Title'] = combine\_df.Name.str.extract(' ([A-Za-z]+)\.', expand=False)

1. 根据现有特征生成新特征（选择数据量充足的特征做新特征，数据量不足的特征不易训练出特性，容易overfit）

combine\_df['High\_Survival\_Ticket'] = np.where(combine\_df['Ticket\_Lett'].isin(['1', '2', 'P']),1,0)

combine\_df['FamilySize'] = np.where(combine\_df['SibSp']+combine\_df['Parch']==0, 'Alone', np.where(combine\_df['SibSp']+combine\_df['Parch']<=3, 'Small', 'Big'))

特征属性分类思路：数据充足->按原分类；数据不足->大于平均值整合一类/小于平均值整合一类

1. Dummy variables

df = pd.get\_dummies(combine\_df['Embarked'],prefix='Embarked')

combine\_df = pd.concat([combine\_df,df],axis=1).drop('Embarked',axis=1)

1. Model, predict and solve

Logistic Regression, SVM, KNN, Random Forest, Decision Tree -> GBDT, XBoost, Naïve Bayes

1. Cross-validation evaluate

X\_train, X\_cv, Y\_train, Y\_cv = train\_test\_split(X\_all, Y\_all, test\_size=num\_test)

logreg.fit(X\_train, Y\_train)

score = logreg.score(X\_cv, Y\_cv) \* 100

1. 查看特征重要性，删除/改进特征重新训练

Logistic regression: coeff\_df["Correlation"] = pd.Series(logreg.coef\_[0])

Random forest: feature\_importance['importance'] = random\_forest.feature\_importances

Learning curve, bad case