1、数据导入

import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import warnings

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import mean\_squared\_log\_error

from math import sqrt

train = pd.read\_csv('input/train.csv')

test = pd.read\_csv('input/test.csv')

sample = pd.read\_csv('input/sample\_submission.csv')

2、数据类型

tmdb = pd.concat([train, test])

tmdb.dtypes.sort\_values()

budget int64

id int64

runtime float64

revenue float64

popularity float64

Keywords object

status object

spoken\_languages object

release\_date object

production\_countries object

production\_companies object

poster\_path object

overview object

original\_title object

original\_language object

imdb\_id object

homepage object

genres object

crew object

cast object

belongs\_to\_collection object

tagline object

title object

dtype: object

3、缺失数据处理

tmdb.isnull().sum()[tmdb.isnull().sum()>0]

Keywords 669

belongs\_to\_collection 5917

cast 26

crew 38

genres 23

homepage 5032

overview 22

poster\_path 2

production\_companies 414

production\_countries 157

release\_date 1

revenue 4398

runtime 6

spoken\_languages 62

status 2

tagline 1460

title 3

dtype: int64

train.belongs\_to\_collection = train.belongs\_to\_collection.fillna("unknow")

test.belongs\_to\_collection = test.belongs\_to\_collection.fillna("unknow")

train.genres = train.genres.fillna("unknow")

test.genres = test.genres.fillna("unknow")

train.homepage = train.homepage.fillna("unknow")

test.homepage = test.homepage.fillna("unknow")

train.overview = train.overview.fillna("unknow")

test.overview = test.overview.fillna("unknow")

train.poster\_path = train.poster\_path.fillna("unknow")

test.poster\_path = test.poster\_path.fillna("unknow")

train.production\_companies = train.production\_companies.fillna("unknow")

test.production\_companies = test.production\_companies.fillna("unknow")

train.production\_countries = train.production\_countries.fillna("unknow")

test.production\_countries = test.production\_countries.fillna("unknow")

train.runtime = train.runtime.fillna(train.runtime.mean())

test.runtime = test.runtime.fillna(train.runtime.mean())

train.spoken\_languages = train.spoken\_languages.fillna("unknow")

test.spoken\_languages = test.spoken\_languages.fillna("unknow")

train.tagline = train.tagline.fillna("unknow")

test.tagline = test.tagline.fillna("unknow")

train.Keywords = train.Keywords.fillna("unknow")

test.Keywords = test.Keywords.fillna("unknow")

train.cast = train.cast.fillna("unknow")

test.cast = test.cast.fillna("unknow")

train.crew = train.crew.fillna("unknow")

test.crew = test.crew.fillna("unknow")

train.title = train.title.fillna("unknow")

test.title = test.title.fillna("unknow")

train.status = train.status.fillna(train.status.mode())

test.status = test.status.fillna(train.status.mode())

train.release\_date = train.release\_date.fillna("unknow")

test.release\_date = test.release\_date.fillna("unknow")

test.release\_date[test.release\_date=="unknow"] = 0000-00-00

d:\python\python36\lib\site-packages\ipykernel\_launcher.py:49: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

4、数据探索及特征工程

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

train['genres2'] = train.genres.apply(lambda x: x.count('id'))

test['genres2'] = test.genres.apply(lambda x: x.count('id'))

train['cast2'] = train.cast.apply(lambda x: x.count('cast\_id'))

test['cast2'] = test.cast.apply(lambda x: x.count('cast\_id'))

train['Keywords2'] = train.Keywords.apply(lambda x: x.count('id'))

test['Keywords2'] = test.Keywords.apply(lambda x: x.count('id'))

train['title2'] = train.title.apply(lambda x: len(x))

test['title2'] = test.title.apply(lambda x: len(x))

train['tagline2'] = train.tagline.apply(lambda x: len(x))

test['tagline2'] = test.tagline.apply(lambda x: len(x))

train['spoken\_languages2'] = train.spoken\_languages.apply(lambda x: x.count('name'))

test['spoken\_languages2'] = test.spoken\_languages.apply(lambda x: x.count('name'))

train['production\_countries2'] = train.production\_countries.apply(lambda x: x.count('name'))

test['production\_countries2'] = test.production\_countries.apply(lambda x: x.count('name'))

train['crew2'] = train.crew.apply(lambda x: x.count('id'))

test['crew2'] = test.crew.apply(lambda x: x.count('id'))

train['overview2'] = train.overview.apply(lambda x: len(x))

test['overview2'] = test.overview.apply(lambda x: len(x))

train['original\_title2'] = train.original\_title.apply(lambda x: len(x))

test['original\_title2'] = test.original\_title.apply(lambda x: len(x))

train['homepage2'] = train.homepage.apply(lambda x: 0 if x=='unknow' else 1)

test['homepage2'] = test.homepage.apply(lambda x: 0 if x=='unknow' else 1)

train['belongs\_to\_collection2'] = train.belongs\_to\_collection.apply(lambda x: 0 if x=='unknow' else 1)

test['belongs\_to\_collection2'] = test.belongs\_to\_collection.apply(lambda x: 0 if x=='unknow' else 1)

train['production\_companies2'] = train.production\_companies.apply(lambda x: x.count('id'))

test['production\_companies2'] = test.production\_companies.apply(lambda x: x.count('id'))

train['release\_date\_year'] = train.release\_date.apply(lambda x: pd.to\_datetime(x).year)

train['release\_date\_month'] = train.release\_date.apply(lambda x: pd.to\_datetime(x).month)

train['release\_date\_day'] = train.release\_date.apply(lambda x: pd.to\_datetime(x).day)

train['release\_date\_weekday'] = train.release\_date.apply(lambda x: pd.to\_datetime(x).weekday())

test['release\_date\_year'] = test.release\_date.apply(lambda x: pd.to\_datetime(x).year)

test['release\_date\_month'] = test.release\_date.apply(lambda x: pd.to\_datetime(x).month)

test['release\_date\_day'] = test.release\_date.apply(lambda x: pd.to\_datetime(x).day)

test['release\_date\_weekday'] = test.release\_date.apply(lambda x: pd.to\_datetime(x).weekday())

train.drop(['id', 'belongs\_to\_collection', 'genres', 'homepage',

'imdb\_id', 'original\_language', 'original\_title', 'overview', 'poster\_path', 'production\_companies',

'production\_countries', 'release\_date', 'spoken\_languages' , 'tagline', 'title', 'Keywords', 'cast', 'crew'], axis=1, inplace=True)

test.drop(['id', 'belongs\_to\_collection', 'genres', 'homepage',

'imdb\_id', 'original\_language', 'original\_title', 'overview', 'poster\_path', 'production\_companies',

'production\_countries', 'release\_date', 'spoken\_languages' , 'tagline', 'title', 'Keywords', 'cast', 'crew'], axis=1, inplace=True)

5、训练数据准备

tmdb = pd.concat([train, test], sort=False)

tmdb = pd.get\_dummies(tmdb)

len\_train = len(train)

len\_test = len(test)

train = tmdb[:len\_train]

test = tmdb[len\_train:]

train.revenue = train.revenue.astype('int')

xtrain=train.drop("revenue", axis=1)

ytrain=train['revenue']

xtest=test.drop("revenue", axis=1)

d:\python\python36\lib\site-packages\pandas\core\generic.py:5096: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row\_indexer,col\_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy

self[name] = value

train.columns

Index(['budget', 'popularity', 'runtime', 'revenue', 'genres2', 'cast2',

'Keywords2', 'title2', 'tagline2', 'spoken\_languages2',

'production\_countries2', 'crew2', 'overview2', 'original\_title2',

'homepage2', 'belongs\_to\_collection2', 'production\_companies2',

'release\_date\_year', 'release\_date\_month', 'release\_date\_day',

'release\_date\_weekday', 'status\_Post Production', 'status\_Released',

'status\_Rumored'],

dtype='object')

plt.figure(figsize=[12, 12])

plt.subplot(3, 3, 1)

sns.barplot('budget', 'revenue', data=train)

plt.subplot(3, 3, 2)

sns.barplot('popularity', 'revenue', data=train)

plt.subplot(3, 3, 3)

sns.barplot('runtime', 'revenue', data=train)

plt.subplot(3, 3, 4)

sns.barplot('popularity', 'revenue', data=train)

plt.subplot(3, 3, 5)

sns.barplot('genres2', 'revenue', data=train)

plt.subplot(3, 3, 6)

sns.barplot('cast2', 'revenue', data=train)

plt.subplot(3, 3, 7)

sns.barplot('Keywords2', 'revenue', data=train)

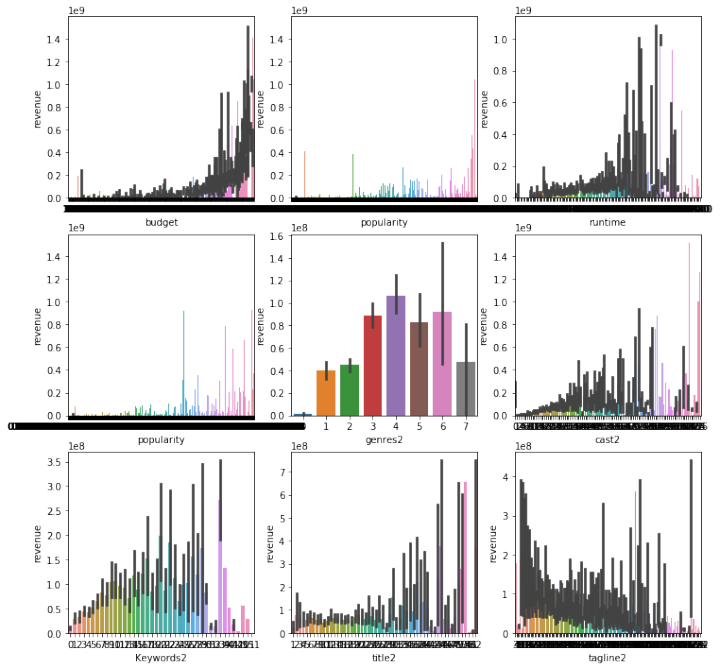
plt.subplot(3, 3, 8)

sns.barplot('title2', 'revenue', data=train)

plt.subplot(3, 3, 9)

sns.barplot('tagline2', 'revenue', data=train)

<matplotlib.axes.\_subplots.AxesSubplot at 0x19650563d30>



plt.figure(figsize=[12, 12])

plt.subplot(3, 3, 1)

sns.barplot('spoken\_languages2', 'revenue', data=train)

plt.subplot(3, 3, 2)

sns.barplot('production\_countries2', 'revenue', data=train)

plt.subplot(3, 3, 3)

sns.barplot('crew2', 'revenue', data=train)

plt.subplot(3, 3, 4)

sns.barplot('overview2', 'revenue', data=train)

plt.subplot(3, 3, 5)

sns.barplot('original\_title2', 'revenue', data=train)

plt.subplot(3, 3, 6)

sns.barplot('homepage2', 'revenue', data=train)

plt.subplot(3, 3, 7)

sns.barplot('belongs\_to\_collection2', 'revenue', data=train)

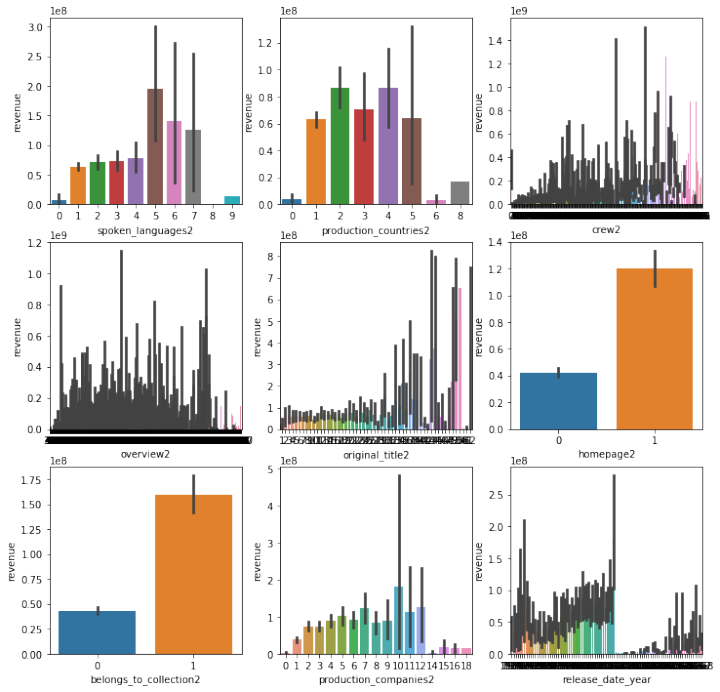
plt.subplot(3, 3, 8)

sns.barplot('production\_companies2', 'revenue', data=train)

plt.subplot(3, 3, 9)

sns.barplot('release\_date\_year', 'revenue', data=train)

<matplotlib.axes.\_subplots.AxesSubplot at 0x196598c4e48>



plt.figure(figsize=[12, 12])

plt.subplot(3, 3, 1)

sns.barplot('release\_date\_month', 'revenue', data=train)

plt.subplot(3, 3, 2)

sns.barplot('release\_date\_day', 'revenue', data=train)

plt.subplot(3, 3, 3)

sns.barplot('release\_date\_weekday', 'revenue', data=train)

plt.subplot(3, 3, 4)

sns.barplot('status\_Post Production', 'revenue', data=train)

plt.subplot(3, 3, 5)

sns.barplot('status\_Released', 'revenue', data=train)

plt.subplot(3, 3, 6)

sns.barplot('status\_Rumored', 'revenue', data=train)

<matplotlib.axes.\_subplots.AxesSubplot at 0x1964aecc908>

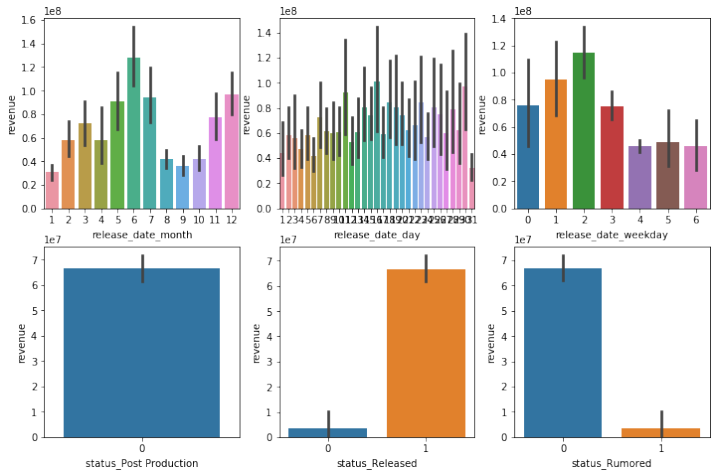
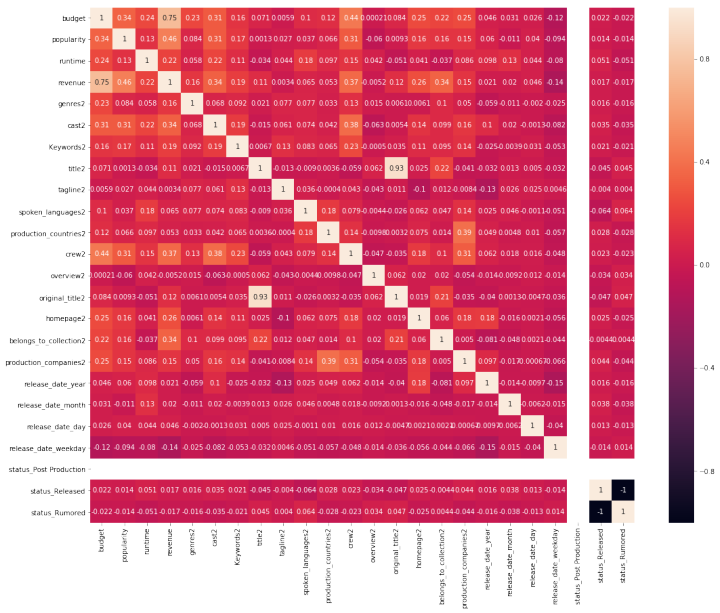


fig = plt.figure(figsize=(18,14))

sns.heatmap(train.corr(), annot=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x1962e78dcf8>



6、模型建立、训练、误差检验

from sklearn import tree

model\_tree = tree.DecisionTreeRegressor()

from sklearn import linear\_model

model\_lr = linear\_model.LinearRegression()

from sklearn import svm

model\_svr = svm.SVR()

from sklearn import neighbors

model\_KN = neighbors.KNeighborsRegressor()

from sklearn import ensemble

model\_RandomForestRegressor = ensemble.RandomForestRegressor(n\_estimators=20)

from sklearn import ensemble

model\_AdaBoostRegressor = ensemble.AdaBoostRegressor(n\_estimators=50)

from sklearn import ensemble

model\_GradientBoostingRegressor = ensemble.GradientBoostingRegressor(n\_estimators=100)

from sklearn.ensemble import BaggingRegressor

model\_BaggingRegressor = BaggingRegressor()

from sklearn.tree import ExtraTreeRegressor

model\_ExtraTreeRegressor = ExtraTreeRegressor()

def try\_model(model):

model\_name = str(model)

train\_X, val\_X, train\_y, val\_y = train\_test\_split(xtrain, ytrain, random\_state=1)

model.fit(train\_X, train\_y)

val\_preds = model.predict(val\_X)

val\_preds[val\_preds<0] = 0

rmsle = np.sqrt(mean\_squared\_log\_error( val\_preds, val\_y))

print("%s - RMSLE: %2f" % (model\_name, sqrt(rmsle)))

model\_list = [model\_tree, model\_lr, model\_svr, model\_KN, model\_RandomForestRegressor,

model\_AdaBoostRegressor, model\_GradientBoostingRegressor, model\_BaggingRegressor,

model\_ExtraTreeRegressor]

for model in model\_list:

try\_model(model)

DecisionTreeRegressor(criterion='mse', max\_depth=None, max\_features=None,

max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

presort=False, random\_state=None, splitter='best') - RMSLE: 1.810177

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None,

normalize=False) - RMSLE: 2.767681

d:\python\python36\lib\site-packages\sklearn\svm\base.py:196: FutureWarning: The default value of gamma will change from 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to avoid this warning.

"avoid this warning.", FutureWarning)

SVR(C=1.0, cache\_size=200, coef0=0.0, degree=3, epsilon=0.1,

gamma='auto\_deprecated', kernel='rbf', max\_iter=-1, shrinking=True,

tol=0.001, verbose=False) - RMSLE: 1.720772

KNeighborsRegressor(algorithm='auto', leaf\_size=30, metric='minkowski',

metric\_params=None, n\_jobs=None, n\_neighbors=5, p=2,

weights='uniform') - RMSLE: 1.616668

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=None,

max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=20, n\_jobs=None,

oob\_score=False, random\_state=None, verbose=0, warm\_start=False) - RMSLE: 1.549074

AdaBoostRegressor(base\_estimator=None, learning\_rate=1.0, loss='linear',

n\_estimators=50, random\_state=None) - RMSLE: 1.986040

GradientBoostingRegressor(alpha=0.9, criterion='friedman\_mse', init=None,

learning\_rate=0.1, loss='ls', max\_depth=3, max\_features=None,

max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1,

min\_samples\_split=2, min\_weight\_fraction\_leaf=0.0,

n\_estimators=100, n\_iter\_no\_change=None, presort='auto',

random\_state=None, subsample=1.0, tol=0.0001,

validation\_fraction=0.1, verbose=0, warm\_start=False) - RMSLE: 2.011112

BaggingRegressor(base\_estimator=None, bootstrap=True,

bootstrap\_features=False, max\_features=1.0, max\_samples=1.0,

n\_estimators=10, n\_jobs=None, oob\_score=False, random\_state=None,

verbose=0, warm\_start=False) - RMSLE: 1.531035

ExtraTreeRegressor(criterion='mse', max\_depth=None, max\_features='auto',

max\_leaf\_nodes=None, min\_impurity\_decrease=0.0,

min\_impurity\_split=None, min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, random\_state=None,

splitter='random') - RMSLE: 1.795049

7、提交结果

test2 = pd.read\_csv('input/test.csv')

model\_BaggingRegressor.fit(xtrain, ytrain)

pred = model\_BaggingRegressor.predict(xtest)

output = pd.DataFrame({'id': test2.id, 'revenue': pred})

output.to\_csv('submission.csv', index=False)