**一、题目介绍**

赛题的目标是预测测试集中二手车的交易价格，总数据量超过40万，包含31列变量信息，其中15列为匿名变量，通过数据科学以及机器学习深度学习的方法来进行建模得到预测结果。该问题属于数据挖掘中的回归问题。

评测标准为MAE(Mean Absolute Error)。

若真实值为，预测值，那么该模型的MAE计算公式为

.

MAE越小，说明模型预测得越准确。

**二、数据分析**

需要使用的库

import **warnings**

**warnings**.**filterwarnings**('ignore')

import **pandas** as **pd**

import **numpy** as **np**

import **matplotlib**.**pyplot** as **plt**

import **seaborn** as **sns**

from **sklearn**.**model\_selection** import **KFold**

from **lightgbm**.**sklearn** import **LGBMRegressor**

from **xgboost**.**sklearn** import **XGBRegressor**

from **sklearn**.**metrics** import **mean\_squared\_error**, **mean\_absolute\_error**

from **sklearn**.**model\_selection** import **GridSearchCV**

导入数据集，并合并训练集和测试集以便于数据清洗。

Train\_data = **pd**.**read\_csv**('Train/used\_car\_train\_20200313.csv', sep=' ')

Test\_data = **pd**.**read\_csv**('Test/used\_car\_testB\_20200421.csv', sep=' ')

df = **pd**.**concat**([Train\_data, Test\_data], ignore\_index=True)

查看数据集的特征的非空数据、数据类型和特征的缺失数量；

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 SaleID 200000 non-null int64

1 name 200000 non-null int64

2 regDate 200000 non-null int64

3 model 199999 non-null float64

4 brand 200000 non-null int64

5 bodyType 193990 non-null float64

6 fuelType 188396 non-null float64

7 gearbox 192051 non-null float64

8 power 200000 non-null int64

9 kilometer 200000 non-null float64

10 notRepairedDamage 200000 non-null object

11 regionCode 200000 non-null int64

12 seller 200000 non-null int64

13 offerType 200000 non-null int64

14 creatDate 200000 non-null int64

15 price 150000 non-null float64

16 v\_0 200000 non-null float64

17 v\_1 200000 non-null float64

18 v\_2 200000 non-null float64

19 v\_3 200000 non-null float64

20 v\_4 200000 non-null float64

21 v\_5 200000 non-null float64

22 v\_6 200000 non-null float64

23 v\_7 200000 non-null float64

24 v\_8 200000 non-null float64

25 v\_9 200000 non-null float64

26 v\_10 200000 non-null float64

27 v\_11 200000 non-null float64

28 v\_12 200000 non-null float64

29 v\_13 200000 non-null float64

30 v\_14 200000 non-null float64

dtypes: float64(21), int64(9), object(1)

None

SaleID 0

name 0

regDate 0

model 1

brand 0

bodyType 6010

fuelType 11604

gearbox 7949

power 0

kilometer 0

notRepairedDamage 0

regionCode 0

seller 0

offerType 0

creatDate 0

price 50000

v\_0 0

v\_1 0

v\_2 0

v\_3 0

v\_4 0

v\_5 0

v\_6 0

v\_7 0

v\_8 0

v\_9 0

v\_10 0

v\_11 0

v\_12 0

v\_13 0

v\_14 0

dtype: int64

其中‘price’为长尾分布，经转换后与标准正态分布对比，黑色曲线为标准正态分布；

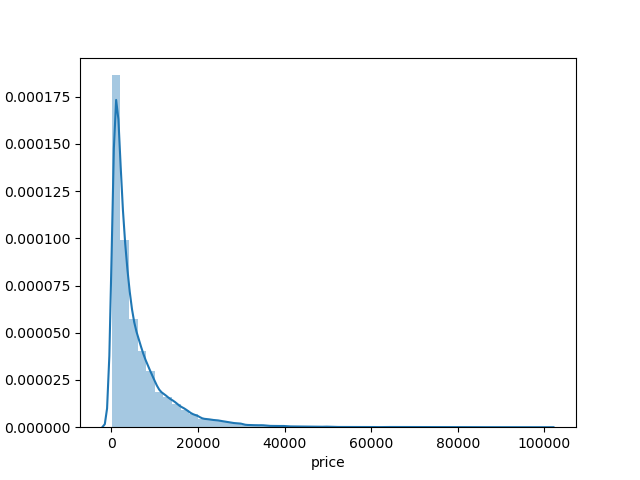


图 1 'price'的分布

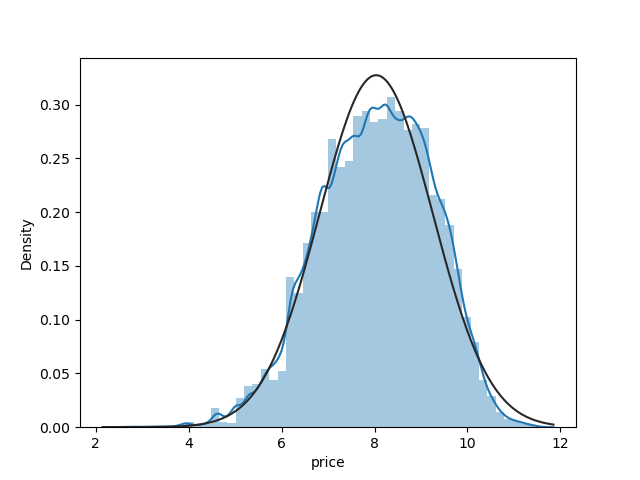


图 2 转换后'price'的分布

观察其他数值特征分布，‘power’的分布比较特别，后面需要做处理；

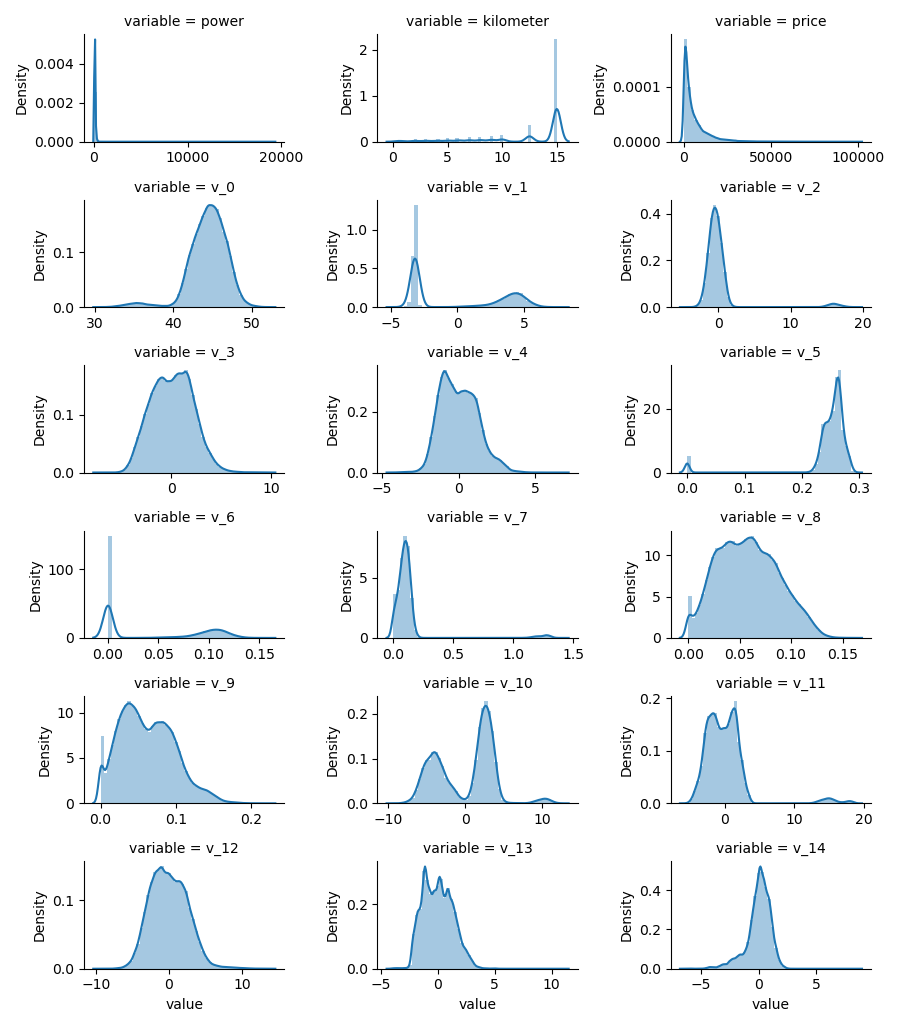


图 3 数值特征的分布

由热力图可知，匿名特征‘v\_0’，‘v\_3’，‘v\_8’，‘v\_12’与‘price’相关性很高。

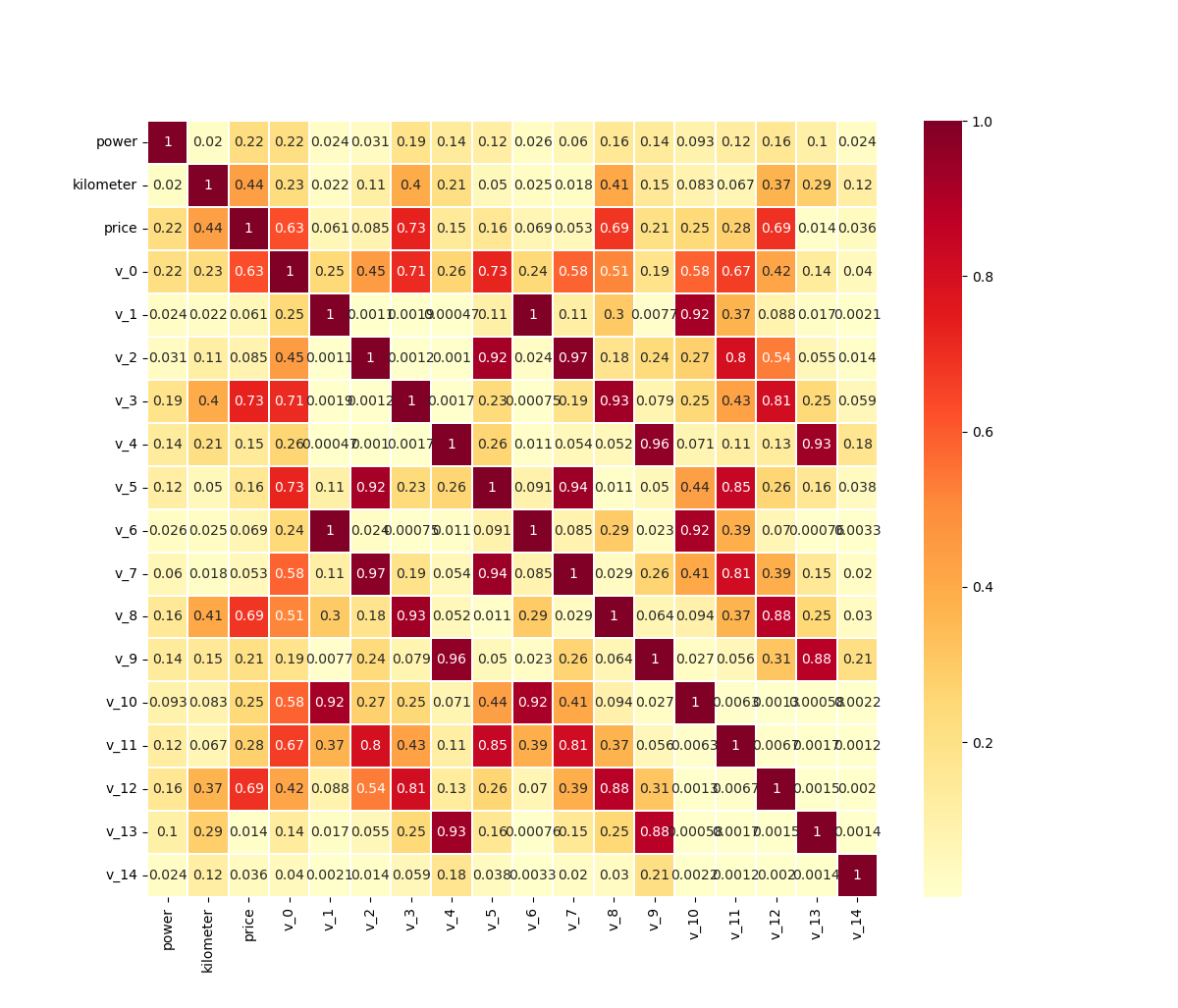


图 4 热力图

**三、数据挖掘**

**1、数据清洗**

对不同特征进行处理，统计‘name’的重复值，删除无关特征，对‘price’进行对数变换；用众数填充缺失值；处理‘power’异常值，并对‘notRepairedDamage’类型转换；对可分类的连续特征进行分桶。

*#统计'name'重复值*

df['name\_count'] = df.**groupby**(['name'])['SaleID'].**transform**('count')

del df['name']

del df['offerType']

del df['seller']

*#对'price'做对数变换*

df['price'] = **np**.log1p(df['price'])

*#用众数填充缺失值*

df['fuelType'] = df['fuelType'].**fillna**(0)

df['gearbox'] = df['gearbox'].**fillna**(0)

df['bodyType'] = df['bodyType'].**fillna**(0)

df['model'] = df['model'].**fillna**(0)

*#处理异常值*

df['power'] = df['power'].**map**(lambda x: 600 if x>600 else x) *#限定power<=600*

df['notRepairedDamage'] = df['notRepairedDamage'].**astype**('str').**apply**(lambda x: x if x != '-' else None).astype('float32') *#类型转换*

*#对可分类的连续特征进行分桶*

**bin** = [i\*10 for i in **range**(31)]

df['power\_bin'] = **pd**.**cut**(df['power'], **bin**, labels=False)

**bin** = [i\*10 for i in **range**(24)]

df['model\_bin'] = **pd**.**cut**(df['model'], **bin**, labels=False)

**2、特征工程**

提取出年、月、日和使用时间；

from **datetime** import **datetime**

def **date\_process**(x):

    year = **int**(**str**(x)[:4])

    month = **int**(**str**(x)[4:6])

    day = **int**(**str**(x)[6:8])

    if month < 1:

        month = 1

    date = **datetime**(year, month, day)

    return date

df['regDate'] = df['regDate'].**apply**(**date\_process**)

df['creatDate'] = df['creatDate'].**apply**(**date\_process**)

df['regDate\_year'] = df['regDate'].dt.year

df['regDate\_month'] = df['regDate'].dt.month

df['regDate\_day'] = df['regDate'].dt.day

df['creatDate\_year'] = df['creatDate'].dt.year

df['creatDate\_month'] = df['creatDate'].dt.month

df['creatDate\_day'] = df['creatDate'].dt.day

df['car\_age\_day'] = (df['creatDate'] - df['regDate']).dt.days*#二手车使用天数*

df['car\_age\_year'] = **round**(df['car\_age\_day'] / 365, 1)*#二手车使用年数*

类别特征对价格的统计最大、最小、平均值等；

car\_cols = ['brand','model','kilometer','fuelType','bodyType']

for col in car\_cols:

    t = Train\_data.**groupby**(col,as\_index=False)['price'].**agg**(

        {col+'\_count':'count',col+'\_price\_max':'max',col+'\_price\_median':'median',

         col+'\_price\_min':'min',col+'\_price\_sum':'sum',col+'\_price\_std':'std',col+'\_price\_mean':'mean'})

    df = **pd**.**merge**(df,t,on=col,how='left')

行驶路程与功率统计；

kp = ['kilometer','power']

t1 = Train\_data.**groupby**(kp[0],as\_index=False)[kp[1]].**agg**(

        {kp[0]+'\_'+kp[1]+'\_count':'count',kp[0]+'\_'+kp[1]+'\_max':'max',kp[0]+'\_'+kp[1]+'\_median':'median',

         kp[0]+'\_'+kp[1]+'\_min':'min',kp[0]+'\_'+kp[1]+'\_sum':'sum',kp[0]+'\_'+kp[1]+'\_std':'std',kp[0]+'\_'+kp[1]+'\_mean':'mean'})

df = **pd**.**merge**(df,t1,on=kp[0],how='left')

根据数据分析阶段结果，v\_0,v\_3,v\_8,v\_12，kilometer与price的相关性很高，进行简单的组合，获取新的特征。

num\_cols = [0,3,8,12]

for i in num\_cols:

    for j in num\_cols:

        df['new'+**str**(i)+'\*'+**str**(j)]=df['v\_'+**str**(i)]\*df['v\_'+**str**(j)]

for i in num\_cols:

    for j in num\_cols:

        df['new'+**str**(i)+'+'+**str**(j)]=df['v\_'+**str**(i)]+df['v\_'+**str**(j)]

for i in num\_cols:

    for j in num\_cols:

        df['new'+**str**(i)+'-'+**str**(j)]=df['v\_'+**str**(i)]-df['v\_'+**str**(j)]

for i in **range**(15):

    df['new'+**str**(i)+'\*year']=df['v\_'+**str**(i)] \* df['car\_age\_year']

**3、建模调参**

**划分训练数据集和测试数据集**

df1 = df.**copy**()

test = df1[df1['price'].**isnull**()]

X\_train = df1[df1['price'].**notnull**()].**drop**(['price','regDate','creatDate','SaleID','regionCode'],axis=1)

Y\_train = df1[df1['price'].**notnull**()]['price']

X\_test = df1[df1['price'].**isnull**()].**drop**(['price','regDate','creatDate','SaleID','regionCode'],axis=1)

**print**("test information:")

test.**info**()

**print**("X\_train information:")

X\_train.**info**()

cols = **list**(X\_train)

oof = **np**.**zeros**(X\_train.shape[0])

sub = test[['SaleID']].**copy**()

sub['price'] = 0

feat\_df = **pd**.**DataFrame**({'feat': cols, 'imp': 0})

skf = **KFold**(n\_splits=10, shuffle=True, random\_state=2022)

训练LightGBM模型

clf = **LGBMRegressor**(

    n\_estimators=10000,

    learning\_rate=0.02,

    boosting\_type= 'gbdt',

    objective = 'regression\_l1',

    max\_depth = -1,

    num\_leaves=31,

    min\_child\_samples = 20,

    feature\_fraction = 0.8,

    bagging\_freq = 1,

    bagging\_fraction = 0.8,

    lambda\_l2 = 2,

    random\_state=2022,

    metric='mae',

    device = 'gpu'  *#不使用gpu这个也注释掉*

    )

lgb\_mae = 0

sub\_lgb = 0

for i, (trn\_idx, val\_idx) in **enumerate**(skf.**split**(X\_train, Y\_train)):

**print**('--------------------- {} fold ---------------------'.**format**(i+1))

    trn\_x, trn\_y = X\_train.iloc[trn\_idx].reset\_index(drop=True), Y\_train[trn\_idx]

    val\_x, val\_y = X\_train.iloc[val\_idx].reset\_index(drop=True), Y\_train[val\_idx]

    clf.**fit**(

        trn\_x, trn\_y,

        eval\_set=[(val\_x, val\_y)],

        eval\_metric='mae',

        early\_stopping\_rounds=300,

*#verbose\_eval=300*

        verbose=False

    )

    sub\_lgb += **np**.expm1(clf.**predict**(X\_test)) / skf.n\_splits

    val\_lgb = clf.**predict**(val\_x)

*#print('val mae:', mean\_absolute\_error(np.expm1(val\_y), np.expm1(val\_lgb)))*

    lgb\_mae += **mean\_absolute\_error**(**np**.expm1(val\_y), **np**.expm1(val\_lgb))/skf.n\_splits

**print**('MAE of val with lgb:', lgb\_mae)

训练XGBoost模型

xlf= **XGBRegressor**(

    tree\_method='gpu\_hist',

    gpu\_id='0',

    n\_estimators=1000,

    gamma=0, subsample=0.8,

    colsample\_bytree=0.9,

    max\_depth=7

    )*#, objective ='reg:squarederror'*

param\_grid = {'learning\_rate': [0.01, 0.05, 0.1, 0.2]}

gbm = **GridSearchCV**(xlf, param\_grid)

xgb\_mae = 0

sub\_xgb = 0

for i, (trn\_idx, val\_idx) in **enumerate**(skf.**split**(X\_train, Y\_train)):

**print**('--------------------- {} fold ---------------------'.**format**(i+1))

    trn\_x, trn\_y = X\_train.iloc[trn\_idx].reset\_index(drop=True), Y\_train[trn\_idx]

    val\_x, val\_y = X\_train.iloc[val\_idx].reset\_index(drop=True), Y\_train[val\_idx]

    gbm.**fit**(

        trn\_x, trn\_y,

        eval\_set=[(val\_x, val\_y)],

        eval\_metric='mae',

        early\_stopping\_rounds=300,

        verbose=False

    )

    sub\_xgb += **np**.expm1(gbm.**predict**(X\_test)) / skf.n\_splits

    val\_xgb= gbm.**predict**(val\_x)

*#print('val mae:', mean\_absolute\_error(np.expm1(val\_y), np.expm1(val\_xgb)))*

    xgb\_mae += **mean\_absolute\_error**(**np**.expm1(val\_y), **np**.expm1(val\_xgb))/skf.n\_splits

**print**('MAE of val with xgb:', xgb\_mae)

**4、模型融合**

使用加权融合的方法对两个模型的数据进行合并，并预测测试集。其中预测结果中存在负数，与实际情况不符，进行简单的修正。最后生成提交文件。

val\_Weighted = (1-lgb\_mae/(lgb\_mae+xgb\_mae))\***np**.expm1(val\_lgb)+(1-xgb\_mae/(lgb\_mae+xgb\_mae))\***np**.expm1(val\_xgb)

val\_Weighted[val\_Weighted<0]=10

**print**('MAE of val with Weighted ensemble:',**mean\_absolute\_error**(**np**.expm1(val\_y),val\_Weighted))

sub\_Weighted = (1-lgb\_mae/(lgb\_mae+xgb\_mae))\*sub\_lgb+(1-xgb\_mae/(lgb\_mae+xgb\_mae))\*sub\_xgb

sub['price'] = sub\_Weighted

sub.**to\_csv**('./sumbit.csv',index=False)

**四、成绩**

分数（score）越小排名越高，长期赛第500名成绩为436.96。



图 5 得分

**参考：**

Datawhale 零基础入门数据挖掘-Baseline, <https://tianchi.aliyun.com/notebook/95422>