特征⼯程：缺失值填充总结（众数,中数,KNN近邻填充,预测填充）

⾯试不仅仅是⼀个找⼯作的过程，还是⼀个向⾯试官交流学习的过程。之前的某次⾯试中，聊到了缺失值填充⽅法，经⾯试官指点学到了⼀些技能，下⾯简要总结⼀下。

常见的缺失值填充⽅法有填充默认值、均值、众数、KNN填充、以及把缺失值作为新的label通过模型来预测等⽅式，为了介绍这⼏种填充⽅法的使⽤以及填充效果，本⽂将在真实数据集上进⾏简单⽐较。

1. 数据集介绍

数据集来源于 。该数据集共有1000条数据，特征共83维，加上id和label共85列，每维特征缺失数量范围为0~911。为了简单⽐较各种填充⽅法的效果，我们选取最简单的⼆分类模型（逻辑回归），选取F1 score作为评测指标。

读取数据集代码如下：

1.

train\_data = pd.read\_csv('train\_data.csv', encoding='gbk')

# 读取数据集

3.filter\_feature = ['id','label']

# 过滤⽆⽤的维度

4.features = []

5.for x in train\_data.columns:

# 取特征

if x not in filter\_feature:

features.append(x)

train\_data\_x = train\_data[features]

train\_data\_y = train\_data['label']

# 划分训练集、测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(train\_data\_x, train\_data\_y, random\_state=1)

2. 常见的填充⽅法

（1）填充固定值

选取某个固定值/默认值填充缺失值。

train\_data.fillna(0, inplace=True) # 填充 0

（2）填充均值

对每⼀列的缺失值，填充当列的均值。

train\_data.fillna(train\_data.mean(),inplace=True) # 填充均值

（3）填充中位数

对每⼀列的缺失值，填充当列的中位数。

train\_data.fillna(train\_data.median(),inplace=True) # 填充中位数

（4）填充众数

对每⼀列的缺失值，填充当列的众数。由于存在某列缺失值过多，众数为nan的情况，因此这⾥取的是每列删除掉nan值后的众数。

1.

train\_data.fillna(train\_data.mode(),inplace=True)

# 填充众数,该数据缺失太多众数出现为nan的情况

2.

features\_mode = {}

for f in features:

print f,

':',

list(train\_data[f].dropna().mode().values) 5.

features\_mode[f] =list(train\_data[f].dropna().mode().values)[0]

train\_data.fillna(features\_mode,inplace=True)

（5）填充上下条的数据

对每⼀条数据的缺失值，填充其上下条数据的值。

train\_data.fillna(method='pad', inplace=True)

# 填充前⼀条数据的值，但是前⼀条也不⼀定有值

train\_data.fillna(0, inplace=True)

train\_data.fillna(method='bfill', inplace=True)

# 填充后⼀条数据的值，但是后⼀条也不⼀定有值

train\_data.fillna(0, inplace=True)

（6）填充插值得到的数据

⽤插值法拟合出缺失的数据，然后进⾏填充。

for f in features:

for f in features:

# 插值法填充

train\_data[f] = train\_data[f].interpolate()

train\_data.dropna(inplace=True)

（7）填充KNN数据

填充近邻的数据，先利⽤knn计算临近的k个数据，然后填充他们的均值。（）除了knn填充，fancyimpute还提供了其他填充⽅法。

from fancyimputeimport KNN

train\_data\_x = pd.DataFrame(KNN(k=

6).fit\_transform(train\_data\_x), columns=features)

（8）填充模型预测的值

把缺失值作为新的label，建⽴模型得到预测值，然后进⾏填充。这⾥选择某个缺失值数量适当的特征采⽤随机森林RF进⾏拟合，其他缺失特征采⽤均值进⾏填充。

new\_label ='SNP46'

new\_features = []

for fin features:

if f != new\_label: 5.

new\_features.append(f)

new\_train\_x = train\_data[train\_data[new\_label].isnull()== False][new\_features]

new\_train\_x.fillna(new\_train\_x.mean(), inplace= True)

# 其他列填充均值

new\_train\_y = train\_data[train\_data[new\_label].isnull()== False][new\_label]

new\_predict\_x = train\_data[train\_data[new\_label].isnull()== True][new\_features]

new\_predict\_x.fillna(new\_predict\_x.mean(), inplace= True)

# 其他列填充均值

new\_predict\_y = train\_data[train\_data[new\_label].isnull()== True][new\_label]

rfr = RandomForestRegressor(random\_state=

666, n\_estimators=10, n\_jobs=-1)

rfr.fit(new\_train\_x, new\_train\_y)

new\_predict\_y = rfr.predict(new\_predict\_x)

new\_predict\_y = pd.DataFrame(new\_predict\_y, columns=[new\_label], index=new\_predict\_x.index) 20.

new\_predict\_y = pd.concat([new\_predict\_x, new\_predict\_y], axis=1)

new\_train\_y = pd.concat([new\_train\_x, new\_train\_y], axis=

new\_train\_y = pd.concat([new\_train\_x, new\_train\_y], axis= 1)

new\_train\_data = pd.concat([new\_predict\_y,new\_train\_y]) 23.

train\_data\_x = new\_train\_data[features]

train\_data\_y = train\_data['label']

3. 实验对⽐

（1）评测指标

选取F1 score进⾏评测。

def countF1(train, predict):

count =

# 统计预测的正确的正样本数

for i in range(len(train)):

if predict[i] ==1

and train[i] ==1:

count +=1

pre = count \*

1.0 / sum(predict)

# 准确率

recall = count \*

1.0 / sum(train)

# 召回率

Return 2 \* pre \* recall / (pre + recall)

（2）对⽐结果

填充⽅式训练集\_F1测试集\_F1

默认值00.705167170.59689922

均值（mean）0.701863350.67768595

中位数(median)0.708268330.67479675

众数(mode)0.704791340.68852459

上⼀个数据（pad）0.704097120.62711864

下⼀个数据（bfill）0.669811320.60169492

插值0.690184050.61333333

KNN\_30.710769230.66393443

KNN\_60.708978330.68852459

KNN\_100.704791340.68032787

随机森林\_feature30.5714285710.4

随机森林\_feature460.5851393190.41509434

（3）实验⼩结

对于缺失值的处理，除了直接删除缺失严重的特征外，还可以选择各种各样的填充⽅法。对于每⼀种填充⽅式⽽⾔，都有其适⽤的场景，没有绝对的好坏之分，因此在做数据预处理时，要多尝试⼏种填充⽅法，选择表现最佳的即可。

参考⽂献

1.