# 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
```

2.

3.

```
filter_feature = [
'id',
'label']
```

'label']
# 过滤无用的维度

4.

features = []

5.

for x
in train_data.columns:
# 取特征

6.

if x
not
in filter_feature:

7.

```
        features.append(x)
```

8.

9.

```
train_data_x = train_data[features]
```

10.

```
train_data_y = train_data[
'label']
```

11.

```
X_train, X_test, y_train, y_test = train_test_split(train_data_x, train_data_y, random_state=
1)
# 划分训练集、测试集
```

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 = {}
```

3.

```
for f in features:
```

4.

```
print f,
':',
list(train_data[f].dropna().mode().values)
```

5.

```
    features_mode[f] =
list(train_data[f].dropna().mode().values)[
0]
```

6.

```
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。

1.

```
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
```

2.

```
train_data.fillna(
0, inplace=
True)
```

3.

4.

```
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
```

5.

```
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。

1.

```
for f in features:
```

```
# 插值法填充
```

2.

```
train_data[f] = train_data[f].interpolate()
```

3.

4.

```
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。

1.

```
from fancyimpute
import KNN
```

2.

3.

```
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。

1.

new_label =
'SNP46'

2.

new_features = []

3.

for f
in features:

4.

if f != new_label:

5.

```
        new_features.append(f)
```

6.

7.

```
new_train_x = train_data[train_data[new_label].isnull()==
False][new_features]
```

8.

```
new_train_x.fillna(new_train_x.mean(), inplace=
True)
# 其他列填充均值
```

9.

```
new_train_y = train_data[train_data[new_label].isnull()==
False][new_label]
```

10.

11.

```
new_predict_x = train_data[train_data[new_label].isnull()==
True][new_features]
```

12.

```
new_predict_x.fillna(new_predict_x.mean(), inplace=
True)
# 其他列填充均值
```

13.

```
new_predict_y = train_data[train_data[new_label].isnull()==
True][new_label]
```

14.

15.

```
rfr = RandomForestRegressor(random_state=
666, n_estimators=
10, n_jobs=
-1)
```

16.

```
rfr.fit(new_train_x, new_train_y)
```

17.

```
new_predict_y = rfr.predict(new_predict_x)
```

18.

19.

```
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)
```

21.

```
new_train_y = pd.concat([new_train_x, new_train_y], axis=
1)
```

22.

```
new_train_data = pd.concat([new_predict_y,new_train_y])
```

23.

24.

```
train_data_x = new_train_data[features]
```

25.

```
train_data_y = train_data[
'label']
```

3. 实验对比

（1）评测指标

选取F1 score进行评测。

1.

```
def countF1(train, predict):
```

2.

```
    count =
0
# 统计预测的正确的正样本数
```

3.

```
for i
in range(len(train)):
```

4.

```
if predict[i] ==
1
and train[i] ==
1:
```

5.

```
        count +=
1
```

6.

```
    pre =  count *
1.0 / sum(predict)
# 准确率
```

7.

```
    recall =  count *
1.0 / sum(train)
# 召回率
```

8.

```
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0 | 0.70516717 | 0.59689922 |
| 均值（mean） | 0.70186335 | 0.67768595 |
| 中位数(median) | 0.70826833 | 0.67479675 |
| 众数(mode) | 0.70479134 | 0.68852459 |
| 上一个数据（pad） | 0.70409712 | 0.62711864 |
| 下一个数据（bfill） | 0.66981132 | 0.60169492 |
| 插值 | 0.69018405 | 0.61333333 |
| KNN_3 | 0.71076923 | 0.66393443 |
| KNN_6 | 0.70897833 | 0.68852459 |
| KNN_10 | 0.70479134 | 0.68032787 |
| 随机森林_feature3 | 0.571428571 | 0.4 |
| 随机森林_feature46 | 0.585139319 | 0.41509434 |

（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.