代码片段

一、 Python

1.1 筛选列表

```
columns_to_drop = missing_percentage[missing_percentage > 30].index
```

二、 Pytorch

2.1 线性回归

```
# 模型定义
class Lr(torch.nn.Module):
    def __init__(self):
        super(Lr, self).__init__()
        self.linear = torch.nn.Linear(X_train_tensor.shape[1], 1)

    def forward(self, x):
        out = self.linear(x)
        return out

# 训练
model = Lr() #1. 实例化模型
criterion = torch.nn.MSELoss() #2. 实例化损失函数
optimizer = torch.optim.SGD(model.parameters(), lr=1e-9) #3. 实例化优化器类
for i in range(200):
    y_predict = model(X_train_tensor) #4. 向前计算预测值
    print(y_predict)
    loss = criterion(Y_train_tensor, y_predict) #5. 调用损失函数传入真实值和预
测值，得到损失结果
    optimizer.zero_grad() #5. 当前循环参数梯度置为0
    loss.backward() #6. 计算梯度
    optimizer.step()  #7. 更新参数的值

print('w=',model.linear.weight)
print('b=',model.linear.bias)

# 测试
y_predict = model(X_test_tensor)

print(f"{model}")
print(f"RMSE                                                   is
{np.sqrt(sm.mean_squared_error(Y_test_tensor.detach().numpy(),
y_predict.detach().numpy()))}")
print(f"MAE   is   {sm.mean_absolute_error(Y_test_tensor.detach().numpy(),
y_predict.detach().numpy())}")
print(f"MSE   is   {sm.mean_squared_error(Y_test_tensor.detach().numpy(),
y_predict.detach().numpy())}")
print(f"R2        is        {sm.r2_score(Y_test_tensor.detach().numpy(),
y_predict.detach().numpy())}")
```

三、 Pandas

3.1 独热编码

```
df_encoded = pd.get_dummies(df_dropped_low_high, columns=list(df_dropped_low_high.
print("\n'独热编码:")
print(df_encoded)
```

3.2 查缺失数据在每列中的比例

```
missing_values = df_train.isnull().sum()

missing_values = missing_values[missing_values > 0]

missing_percentage = (missing_values / len(df_train)) * 100

print(missing_percentage)
```

3.3 填充缺失值

```
for column in columns_to_fill:
     if df_train[column].dtype == "float64" or df_train[column].dtype ==
"int64":
        df_train[column].fillna(df_train[column].mean(), inplace=True)
    else:
        df_train[column].fillna(df_train[column].mode()[0], inplace=True)

df_train.head()
```