In [1]:
```python
# 随机森林例子
from sklearn.datasets import load_boston
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestRegressor

boston = load_boston()
rfr = RandomForestRegressor(n_estimators=100, random_state=0)
cross_val_score(rfr, boston.data, boston.target, cv=10
                ,scoring="neg_mean_squared_error")
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Functi
on load_boston is deprecated; `load_boston` is deprecated in 1.0 and will be removed in 1.2.

    The Boston housing prices dataset has an ethical problem. You can refer to
    the documentation of this function for further details.

    The scikit-learn maintainers therefore strongly discourage the use of this
    dataset unless the purpose of the code is to study and educate about
    ethical issues in data science and machine learning.

    In this special case, you can fetch the dataset from the original
    source::

        import pandas as pd
        import numpy as np


        data_url = "http://lib.stat.cmu.edu/datasets/boston"
        raw_df = pd.read_csv(data_url, sep="\s+", skiprows=22, header=None)
        data = np.hstack([raw_df.values[::2, :], raw_df.values[1::2, :2]])
        target = raw_df.values[1::2, 2]

    Alternative datasets include the California housing dataset (i.e.
    :func:`~sklearn.datasets.fetch_california_housing`) and the Ames housing
    dataset. You can load the datasets as follows::

        from sklearn.datasets import fetch_california_housing
        housing = fetch_california_housing()

    for the California housing dataset and::

        from sklearn.datasets import fetch_openml
        housing = fetch_openml(name="house_prices", as_frame=True)

    for the Ames housing dataset.

  warnings.warn(msg, category=FutureWarning)
```

Out[1]:
```
array([-11.22504076,  -5.3945749 ,  -4.74755867, -22.54699078,
       -12.31243335, -17.18030718,  -6.94019868, -94.14567212,
       -28.541145  , -14.6250416 ])
```

In [2]:
```python
# 导入库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.impute import SimpleImputer
```

In [3]:
```python
# 以波士顿数据集为例，导入完整的数据集并探索
n_samples, n_features = boston.data.shape
print(n_samples)
print(n_features)
```

```
506
13
```

In [4]:
```python
# 为完整数据集放入缺失值
rng = np.random.RandomState(0)
missing_rate = 0.5
n_missing_samples = int(np.floor(n_samples * n_features * missing_rate))
n_missing_samples
```

Out[4]: 3289

In [5]:
```python
X_missing = boston.data.copy()

missing_samples = rng.randint(0, n_samples, n_missing_samples)
missing_features = rng.randint(0, n_features, n_missing_samples)

X_missing[missing_samples, missing_features] = np.nan
X_missing = pd.DataFrame(X_missing)
```

In [6]:
```python
# 使用0和均值填补缺失值
imp_mean = SimpleImputer(missing_values=np.nan, strategy='mean')
X_missing_mean = imp_mean.fit_transform(X_missing)

imp_0 = SimpleImputer(missing_values=np.nan, strategy='constant', fill_value=0)
X_missing_0 = imp_0.fit_transform(X_missing)
```

In [7]:
```python
# 使用随机森林填补缺失值
X_missing_reg = X_missing.copy()
sortindex = np.argsort(X_missing_reg.isnull().sum(axis=0)).values
y = boston.target

print(pd.concat([X_missing_reg.isnull().sum(axis=0),np.argsort(X_missing_reg.isnull().sum(ax
is=0))], axis=1))

for i in sortindex:

  # 构建新新标签和新数据集
  df = X_missing_reg
  fillc = df.iloc[:,i]
  df = pd.concat([df.iloc[:,df.columns!=i], pd.DataFrame(y)], axis=1)

  # 填补新数据集
  df_0 = imp_0.fit_transform(df)

  # 拆分训练集和测试集
  Ytrain = fillc[fillc.notnull()]
  Ytest = fillc[fillc.isnull()]
  Xtrain = df_0[fillc.notnull(),:]
  Xtest = df_0[fillc.isnull(),:]

  # 用随机森林回归来填补缺失值
  rfr = RandomForestRegressor(n_estimators=100)
  rfr = rfr.fit(Xtrain, Ytrain)
  Ypredict = rfr.predict(Xtest)
  X_missing_reg.loc[fillc.isnull(),i] = Ypredict
```

```
      0   1
0   200   7
1   193   2
2   189   1
3   196   3
4   202   8
5   206   0
6   213   9
7   182   4
8   199  12
9   200   5
10  206  10
11  211  11
12  203   6
```

In [8]:
```python
# 对填补好的数据进行建模
X = [boston.data, X_missing_mean, X_missing_0, X_missing_reg]
mse = []

for x in X:
  rfr = RandomForestRegressor(random_state=0, n_estimators=100)
  score = cross_val_score(rfr, x, boston.target, cv=10
                          , scoring='neg_mean_squared_error').mean()
  mse.append(score * -1)
```

In [17]:
```python
# 绘出条形图
x_labels = ['Full data',
            'Zero Imputation',
            'Mean Imputation',
            'Regressor Imputation']
colors = ['red', 'green', 'blue', 'orange']

plt.figure(figsize=(12,6))
ax = plt.subplot(111)
for i in np.arange(len(mse)):
  ax.barh(i, mse[i], color=colors[i], alpha=0.6, align='center')
ax.set_title('Imputation Techniques with Boston Data')
ax.set_xlim(left=np.min(mse)*0.9, right=np.max(mse)*1.1)
ax.set_xlabel('MSE')
ax.set_yticks(range(len(mse)))
ax.set_yticklabels(x_labels)
plt.show()
```