亿图脑图 MindMaster

浮动主题

数据挖掘

Numpy

- ndarray
  - 属性
    - ndim — 矩阵的维度
    - shape — 矩阵的形状，如（行数，列数）
    - size — 矩阵中元素的个数
    - dtype — 元素数据类型
    - T — 矩阵的转置
  - 方法（为方便记忆，以二维矩阵为例）
    - reshape(shape) — reshape返回新的数组
    - resize(shape) — resize修改原有的数组
    - sum(), max(), min(), mean() — 可设置axis按行列计算
    - tolist() — array -> list
    - fill(value)
  - 操作
    - 索引 — array[行索引]
    - 切片 — array[起始行:结束行:步长, 起始列:结束列:步长]
- 文件读写
  - np.loadtxt(filename, delimiter=,encoding=)
    - dtype：默认以浮点数类型读取
    - comments：默认忽略掉文件中以#号开头的注释行
    - delimiter：无默认值，应该指定字段分隔符，如','
    - usecols：默认取所有列，可以设置读取指定的列，如(1,3)指取文件的第2和第4列
    - skiprows：默认0，不跳过头行，可指定跳过开头的n行
    - maxrows：默认取文件的所有行，可设置读取文件的最大行数，不包括skiprows
  - np.savetxt(filename, array_data,)
    - fmt：默认以保留6位浮点数保存数据，可设置保留整数%d
    - delimiter：默认以空格分隔字段，可设置csv如','
    - encoding：默认系统编码方式
- 矩阵操作
  - 生成矩阵
    - np.array(object)
    - np.empty(shape), np.empty_like(a)
    - np.zeros(shape), np.zeros_like(a)
    - np.ones(shape), np.ones_like(a)
    - np.full(shape,value), np.full_like(a,value)
    - np.eye(ndim)
    - np.fromstring()
    - np.arange(start,end,step)
    - np.linspace(first,last,size)
    - np.concatanate([a,b],axis=0)
  - np.random.rand(), np.random.randn(), np.random.randint(), np.random.uniform(), np.random.normal(), ...
  - 聚合操作 — np.sum(), np.mean(), np.median(), np.max(), np.min(), np.std(),...
- 基本数据结构
  - SERIES

index	element
0	1
1	2
2	3
3	4
4	5

Series（数据列）

  - 属性
    - .index — 获取行索引对象
    - .values — 获取element数组
    - .shape/.ndim/.size/.dtype — 和numpy数组相同
  - 方法
    - .copy() — 复制一个新对象
    - .tolist() — 转换成列表
    - .get(index[,default]) — 按index取值可以设置缺省值
    - .head()/.tail() — 头/尾n个元素
    - .value_counts() — 统计每个值的个数
    - .map(dict|Series|function) — 值映射时
      - `>> pd.Series(["男","女"]).map({"男":1,"女":0})`
      - `>> pd.Series(["男","女"]).map("我是{}生".format)`
      - `pd.Series([0,1,2]).map(lambda x:x+2)`
    - .apply(function) — 对Series中的每一个元素执行function
      - `pd.Series([0,1,2]).apply(lambda x:x+2)`
  - Data Frame

index	a	b
0	x	x
1	x	x
2	x	x
3	x	x
4	x	x

columns；rows；DataFrame（数据框）

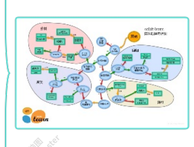
  - 属性
    - .index — 获取行索引对象
    - .values — 获取数据矩阵
    - .columns — 获取列索引对象
    - .T — 获取转置DataFrame
    - .shape/.ndim/.size/.dtype — 和numpy矩阵相同
    - .get(列名, 缺省值) — 获取列，和df[列名]类似，但不会因列名不存在报错
  - 方法
    - .copy() — 复制一个新对象
    - .head(n)/.tail(n) — 头/尾n行
    - .describe() — 数据分布和统计
      - .max()/.min()/.mean()/.median()/.sum()/.....
    - .append(df_or_dict,ignore_index=True) — 添加行
    - df_iris.loc[df_iris.index.max()+1] = [5.5,3.3,1.4,0.2,0] — 添加或修改行
    - .insert(插入位置, 插入列名, 插入数据) — 插入列
    - .drop() — 默认删除行列后生成新的DataFrame，若需在原dataframe上删除可设置参数inplace=True
      - 删行 — .drop(行号)
      - 删列 — .drop(列名, axis=1)
      - 删列，直接作用在df上 — del df[列名]
    - .drop_duplicates(subset=[列名], keep="first|last",inplace=False) — 删除重复项
    - .unique() — 唯一值
    - .set_index(列名) — 使用某列作为index
    - .reset_index(drop=True|False, inplace=True|False) — 重新使用序号作为index
    - .apply(function)
      - df.apply(function)
        - df.apply(np.sum) — 示例：对df每列进行求和 — 对每列数据执行指定的函数运算
      - df.apply(function, axis=1)
        - `df_iris["sepal_area"] = df_iris.apply(lambda x:np.pi*(x["sepal_length"]/2)*(x["sepal_width"]/2), axis=1)` — 示例：根据iris花瓣长和宽，计算近似面积，生成新的一列
        - `df_student["是否成年"] = df_student.apply(lambda x:"是" if x["年龄"]>=18 else "否", axis=1)` — 示例：根据student年龄，判断是否成年，生成新的一列
        - 对每行数据执行指定的函数运算
      - .apply_map(function) — 对每个单元格元素执行指定的函数运算
        - df.applymap(lambda x:"%.2f" % x) — 示例：对所有数值进行格式化保留两位小数
    - pd.get_dummies() — 对离散数据进行One-Hot编码
    - pd.cut() — 将连续数据变成区间类别
      - 按数值划分间隔
        - pd.cut(values, bins=3)
        - pd.cut(values, bins=3)
      - 按频数划分间隔 — pd.qcut()
  - 数据选取
    - 按行索引(index)标签取
      - df.loc[行标签] — 取一行数据到一个Series对象
      - df.loc[[行标签]] — 取一行数据到一个DataFrame对象
      - df.loc[[行标签1,行标签2,...]] — 取多行数据到一个DataFrame对象
      - df.loc[首行标签:末行标签:步长] — 行切片
    - 按行索引(index)位置取
      - df.iloc[行号] — 取一行数据到一个Series对象
      - df.iloc[[行号]] — 取一行数据到一个DataFrame对象
      - df.iloc[[行号1,行号2,...]] — 取多行数据到一个DataFrame对象
      - df.iloc[起始行号:结束行号:步长] — 行切片
    - 取列
      - 按列名取
        - df[列标签] — 取一列数据到一个Series对象
        - df[[列标签]] — 取一列数据到一个DataFrame对象
        - df[[列标签1,列标签2,...]] — 取多列数据到一个DataFrame对象
        - df.loc[:, 首列标签:末列标签:步长] — 列切片
      - 按列号取
        - df.iloc[:, 列号] — 取一列数据到一个Series对象
        - df.iloc[:, [列号]] — 取一列数据到一个DataFrame对象
        - df.iloc[:, [列号1,列号2,...]] — 取多列数据到一个DataFrame对象
        - df.iloc[:, 起始列号:结束列号:步长] — 列切片
    - 取行列
      - df.loc[行标签,列标签] — df.loc[[行标签1,行标签2,...], [列标签1,列标签2,...]]
      - df.loc[首行标签:末行标签:步长, 首列标签:末列标签:步长]
      - df.iloc[位置,位置] — df.iloc[[行号1,行号2,...], [列号1,列号2,...]]
      - df.iloc[起始行号:结束行号:步长, 起始列号:结束列号:步长]
      - 可灵活搭配列表或切片的方法
  - 空值类型
    - np.nan
    - pd.nat
  - 空值检测
    - .isna() — 得到一个元素是否为空的bool矩阵
    - .notna() — 得到一个元素是否不为空的bool矩阵
    - .isna().any() — 查看包含空值的列
    - .isna().any(axis=1) — 查看包含空值的行
    - .isna().all() — 查看全部空值的列
    - .isna().all(axis=1) — 查看全部空值的行
    - .isna().sum() / .isna().sum(axis=1) — 统计每列/每行空值数量
  - 空值填充
    - .fillna(填充值)
      - `df_student["成绩"].fillna(60)` — 示例：使用60填充空值
      - `df_student["成绩"].fillna(df_student["成绩"].mean())` — 示例：使用成绩的平均数填充空值
      - 用合理空值填充空值
    - .fillna(method='ffill') — 用上一个值填充空值
    - .fillna(method='bfill') — 用下一个值填充空值
    - .fillna(method='ffill', axis=1) — 用左一个值填充空值
    - .fillna(method='bfill', axis=1) — 用右一个值填充空值
    - df["成绩"].fillna(df["成绩"].mean()) — 用合理空值填充空值
  - 空值清除
    - .dropna()
      - .dropna() — 默认去除包含任何空值的行
      - .dropna(axis=1) — 去除包含任何空值的列
      - .dropna(how='all') — 只有一行所有值都为空才去除
      - .dropna(subset=包含空值的行或列) — 只判断给定行或列中是否有空值
  - 数据重复
    - 重复检测
      - .duplicated() — 生成一个是否重复行的bool数组
    - 重复去除
      - .drop_duplicates()
        - .drop_duplicates(subset=列名) — 按列名检测重复项
        - drop_duplicates(keep="first"|"last") — 保留第一个或者最后一个
  - 排序
    - 按index排序
      - .sort_index() — 对行索引排序
      - .sort_index(axis=1) — 对列索引(列名)排序
    - 按值排序
      - .sort_values(by) — 按by列值排序
      - .sort_values(by, axis=1) — 按by行值排序
      - `df.sort_values(["成绩","年龄"],ascending=[False,True])` — 多条件排序(先以成绩降序，再以年龄升序)
      - `df.sort_values(["成绩","年龄"],ascending=False)` — 多条件排序(两个条件都降序)
      - 通用参数：
        - ascending：True升序，False降序
        - inplace：是否对原数据
        - na_position："first"|"last" 缺失值排在最前或最后
  - 合并
    - 数据拼接
      - 行拼接，列集合 — pd.concat([df1,df2],axis=0)
      - 列拼接，行集合 — pd.concat([df1,df2],axis=1)
    - 数据关联
      - df1.merge(df2,left_on=左表列名,right_on=右表列名, how='innerleft|right|outer|cross') — 默认使用相同的列进行关联，也可以指定两个df的不同名关联列
      - df1.join(df2,on=列名或列号,how='left|right|inner|outer') — 默认使用index进行关联，也可以指定两个df的公共关联列
  - 分组
    - .groupby()函数
      - .groupby(列名)
        - `df_student.groupby("班级")` — 单字段分组
        - `df_student.groupby(["班级","性别"])` — 多字段分组
      - 使用函数分组
        - 使用函数对行进行分组
          - `def set_index_group(index):`
            `  if index % 2 == 0:`
            `  return "偶数行"`
            `  else:`
            `  return "奇数行"`
            `df_iris.groupby(set_index_group)`
        - 使用函数对列进行分组
          - `def set_column_group(column_name):`
            `  if "length" in column_name:`
            `  return "length"`
            `  elif "width" in column_name:`
            `  return "width"`
            `  else:`
            `  return "classification"`
            `df_iris.groupby(set_column_group,axis=1)`
    - GroupBy对象
      - .groups — 得到所有的group
      - .get_group() — 得到指定的group
      - .apply()
        - df_iris = df_iris.groupby("classification").apply(lambda x:x.fillna(x.mean()))
        - df_iris['sepal_width'] = df_iris.groupby("classification")['sepal_width'].apply(lambda x:x.fillna(x.mean()))
        - 对group数据执行函数
      - .agg()|.aggregate() — 对group进行聚合统计
      - .max()/.min()/.mean()/.median()/.sum()/..... — 对group数据统计
  - 数据导入与导出
    - pd.read_sql_query(sql,conn)
      - 连接关系数据库执行SQL语句并获取结果集
        - 参数：conn=pymysql.connect(host='localhost',port=3306,user='root',passwd='密码',db='数据库名')
        - 参数：conn=sqlite3.connect('mydb.db')
    - 数据导入
      - pd.read_csv(file)
        - 读取列分隔符文件，如CSV
          - 分隔符 — 参数: delimiter|sep=','
          - 注释符 — 参数：comment='#'
          - 文本编码 — 参数：encoding='utf-8'
          - 列头行 — 参数: header=0
          - 指定列名 — 参数: names=[]
          - 指定列作为行索引 — 参数: index_col=[]
          - 读取指定列 — 参数: usecols=[]
          - 跳过数据行数 — 参数: skiprows=0
          - 截止数据行数 — 参数: skipfooter=0
          - 读取数据行数 — 参数: nrows=0
          - 是否转换日期字符串为日期格式 — 参数：parse_dates=False
          - 是否自动设置日期格式 — 参数：infer_datetime_format=False
      - pd.read_excel(file)
        - 读取Excel工作簿，很多参数可以参考read_csv
          - 指定工作表 — 参数：sheet_name=0
      - pd.read_xml(file_or_buffer) — 指定读取XPATH路径 — 参数：xpath="./*"
      - pd.read_html(file_or_buffer) — 读取HTML页面
      - pd.read_json() — 读取JSON字符串或者JSON文件
      - pd.read_pickle() — 读取pickle序列化文件
    - 数据导出
      - df.to_csv() — 将df数据写入分隔符文本文件
      - df.to_excel() — 将df数据写入Excel表格文件
      - df.to_xml() — 将df数据写入XML文件
      - df.to_html() — 将df数据写入HTML文件
      - df.to_json() — 将df数据写入JSON文件
      - df.to_pickle() — 将df数据写入pickle序列化文件
    - 数据转换
      - df.to_dict() — 转换成字典
      - df.to_numpy() — 转换成Numpy数组
      - df.to_string() — 转换成字符串
      - pd.get_dummies()
      - pd.cut()
      - pd.qcut()
  - 数据可视化
    - df.plot() — 绘制折线图
    - df.plot.bar() — 绘制柱状图
    - df.plot.barh() — 绘制横柱图
    - df.plot.scatter() — 绘制散点图
    - df.plot.box() — 绘制箱型图
    - df.plot.hist() — 绘制直方图
    - df.plot.pie() — 绘制饼图

Pandas

- 数据导入与预处理
  - 数据导入
  - 数据去重
  - 数据空值处理
  - 数据转换
- 特征工程
  - 特征选取
    - 特征数据（X）
    - 目标数据（y）
  - 特征构造
    - 特征编码
      - Lable
      - One-Hot
- 数据集划分
  - 监督学习
    - 特征（X）
      - 训练集（X_train）
      - 测试集（X_test）
    - 目标（y）
      - 训练集（y_train）
      - 测试集（y_test）
- 模型选择
  - 监督学习
    - 分类问题：离散值的预测
      - KNN（K-Nearest-Neighbor）
      - NB（Naive-Bayes）
      - LogisticRegression
      - SVM(SVC)
      - DecisionTreeClassifier
    - 回归问题：连续值的预测
      - KNN
      - LR(LinearRegressor)
      - SGD(SGDRegressor)
      - SVM(SVR)
      - DecisionTreeRegressor
  - 非监督学习
    - 聚类
      - 划分聚类 — K均值
      - 层次聚类 — AgglomerativeClustering
      - 密度聚类 — DBSCAN(Density-Based Scan)
    - 降维 — 主成分分析（PCA）
    - 关联分析
- 模型适配
  - fit — 拟合、训练
- 模型预测
  - predict
- 模型评估
  - score
    - 分类问题
      - accuracy_score
      - precision_score
      - recall_score
      - f1_score
    - 回归问题 — r2_score
    - 聚类问题 — 轮廓系数
- 模型保存

sklearn

- datasets
  - load_iris()
  - fetch_...() — sklearn自带数据集
- preprocessing
  - scale() — 标准化方法
  - normalize() — 正则化方法
  - Nomalizer() — 正则化构造器
  - StandarScaler() — 标准化构造器
  - MinMaxScaler()
  - MaxAbsScaler() — 归一化构造器
  - Imputer() — 空值处理器
  - OneHotEncoder() — One-Hot编码器
  - LabelEncoder() — 标签编码器
  - Binarizer() — 二值化编码器
  - PolynomialFeatures() — 多项式构造器
- feature_extraction
  - text
    - TfidfVectorizer — 词频率向量化
    - CountVectorizer — 词频数向量化
- model_selection
  - train_test_split()
  - ShuffleSplit.split()
  - cross_validate_score(estimator,X,y,cv) — 交叉验证
  - KFold
  - GridSearchCV(estimator,param_grid,cv) — 网格搜索
- svm
  - SVC
    - C
    - kernal
      - 'rbf'
      - 'poly' — degree
      - 'linear'
      - 'sigmoid'
    - gamma
  - SVR
  - 支持向量机
- naive_bayes
  - GaussianNB
  - BernulliNB
  - MultinationalNB
  - 朴素贝叶斯
- neighbors
  - KNeighborsClassifier — n_neighbors
  - KNeighborsRegressor — n_neighbors
  - 最近邻
- linear_model
  - LinearRegression
  - SGDRegressor — alpha — 梯度下降学习率
  - 线性回归
  - Ridge — alpha — 正则化系数
  - Lasso — alpha — 正则化系数
  - 带惩罚的线性回归（解决并维带来的过拟合问题）
  - LogisticRegression
    - penalty — 'l1' / 'l2'
    - C
    - solver
      - 'liblinear', 'lbfgs', 'saga' — l1-only
      - 'newton-cg', 'sag' — l2
    - max_iter
    - multi_class — 'ovr', 'multinomial'
  - 线性模型



- tree
  - DecisionTreeClassifier()
    - max_depth
    - criterion — 'gini' / 'entropy'
  - DecisionTreeRegressor()
    - max_depth
    - criterion — 'mse' / 'mae'
  - 决策树
- clusering
  - KMeans(n_clusters) — 划分聚类
  - AgglomerativeClustering(n_clusters) — 层次聚类
  - DBSCAN(eps,min_samles) — 密度聚类
- decomposition
  - PCA(n_components)
    - PCA(3) — 降至3维
    - PCA('mle') — 最大似然估计降维(自动维度)
    - PCA(0.95,svd_solver='full') — 保留95%信息降维
  - 主成分分析
- ensemble
  - Bagging
    - BaggingClassifier(base_estimator,n_estimators...)
    - BaggingRegressor(base_estimator,n_estimators...)
    - RandomForest
      - RandomForestRegressor(n_estimators,max_depth,min_sample_leaf,...)
      - RandomForestClassifier(n_estimators,max_depth,min_sample_leaf,...)
  - Boosting
    - AdaBoosting
      - AdaBoostClassifier(base_estimator,n_estimators...)
      - AdaBoostRegressor(base_estimator,n_estimators...)
    - GBDT
      - GradientBoostingClassifier(base_estimator,n_estimators...)
      - GradientBoostingRegressor(base_estimator,n_estimators...)
      - XGBoost（eXtreme Gradient Boosting）
  - 集成学习
- matrics
  - accuracy_score()
  - precision_score()
  - recall_score()
  - f1_score()
  - classification_report()
  - mean_square_error()

机器学习框架