РЕАЛИЗАЦИЯ МЕТОДИКИ ПО ПОДГОТОВКЕ БОЛЬШИХ ДАННЫХ ДЛЯ ПРОГНОЗНОГО АНАЛИЗА НА ЯЗЫКЕ PYTHON

В.А. Галлингер, студент гр.8ПМ11,
А.В. Семенюта, студент гр.8ПМ11,
Е.И. Губин, к.ф.-м.н., доц.
Томский политехнический университет
E-mail: gubine@tpu.ru

Введение

В предыдущих работах по данной тематике подготовки исходных больших данных для прогнозного анализа использовались программные продукты SAS [1]. Так как в последнее время существенно выросла популярность использования языка Python для больших данных при анализе и построении прогнозных моделей, авторы решили дополнить данный пробел и, основываясь на методических рекомендациях, представленных в работах [2], написать код на языке Python.

Поэтому целью работы является совершенствование реализации методики подготовки исходных данных для построения качественных прогнозных моделей классификации на языке Python.

Описание алгоритма

Для решения поставленной задачи были использованы язык Python версии 3.7.4, IDE JupyterLab версии 1.1.4, а также библиотеки pandas версии 0.25.1, numpy версии 1.18.2 и scikit-learn версии 0.22.2. post1.

В данной работе предложена методика подготовки данных для построения прогнозных моделей классификации с использованием технологий Python. В соответствии с [2], этапы подготовки данных включают в себя следующие шаги:

1. перевод исходного файла во внутренний формат библиотеки pandas DataFrame;
2. проверку исходных данных на ошибки («описки»);
3. проверку исходных данных на пропущенные значения («missings»);
4. проверку исходных данных на выбросы данных («outliers»);
5. проверку исходных данных на наличие дублирующих строк (наблюдений);
6. проверку исходных данных объясняющих переменных (атрибутов) на мультиколлинеарность;

Основные этапы подготовки исходных данных на языке Python

В настоящей работе авторы хотели бы представить методику подготовки исходных («сырых») данных, включающую обязательные шаги статистического анализа данных и организации их формата для корректного предиктивного анализа на языке Python.

Для анализа исходных данных на языке Python показан пример преобразования исходного файла «Rezerv_11.xls» во внутренний формат библиотеки pandas DataFrame.

На рисунке 1 представлен код преобразования.

```
def parse_excel(filepath: str) -> List[pd.core.frame.DataFrame]:
    dfs = []
    xl = pd.ExcelFile(filepath)
    for sheet in xl.sheet_names:
        dfs.append(xl.parse(sheet))
    return dfs
```

Рис. 1. Код функции преобразования табличных данных

В следующем фрагменте на рисунке 2 представлена функция проверки исходных данных на пропущенные значения («missings»).

```
def get_missing_freq(df: pd.core.frame.DataFrame) -> pd.core.frame.DataFrame:
    df_miss_freq = pd.DataFrame()
    df_miss_freq['Not missing'] = df.notna().sum()
    df_miss_freq['Missing'] = df.isna().sum()
    return df_miss_freq
```

Рис. 2. Код функции проверки на пропущенные значения

На рисунке 3 представлена функция проверки исходных данных на выбросы («outliers»).

```
def print_outliers(df: pd.core.frame.DataFrame, feature: str):
    lowest = df.sort_values([feature])[feature][:5]
    highest = df.sort_values([feature])[feature][-5:]
    print(f'The lowest values of {feature} feature:')
    print(lowest)
    print('')
    print(f'The highest values of {feature} feature:')
    print(highest)
```

Рис. 3. Код функции проверки на выбросы

Проверка исходных данных на наличие дублирующих наблюдений (строк) показана рисунке 4 ниже.

```
def print_duplicates(df: pd.core.frame.DataFrame):
    index_list = list(df[df.duplicated()].index)
    count = df.duplicated().sum()
    print(f'Amount of duplicated rows: {count}\nlist of indexes: {index_list}')


def drop_duplicates(df: pd.core.frame.DataFrame) -> pd.core.frame.DataFrame:
    return df.drop_duplicates()
```

Рис. 4. Код функции проверки на наличие дубликатов

Проверка исходных данных объясняющих переменных на наличие мультиколлинеарности представлена на рисунке 5.

```
def get_stats_corr_table(df: pd.core.frame.DataFrame) -> Tuple[pd.core.frame.DataFrame]:
    return df.describe(), df.corr()

def print_corr_columns(df: pd.core.frame.DataFrame, corr_treshold: float = 0.9):
    df_corr = df.corr()
    corr_cols = []
    columns = list(df_corr.columns)
    for i in range(len(columns)):
        j = i + 1
        while j < len(columns):
            if df_corr[columns[i]][columns[j]] > corr_treshold:
                corr_cols.append(columns[i])
                corr_cols.append(columns[j])
            j += 1
    print(corr_cols)


def del_feature(df: pd.core.frame.DataFrame, feature: str) -> pd.core.frame.DataFrame:
    return df.drop(feature, axis=1)
```

Рис. 5. Код функции проверки на наличие мультиколлинеарности

Заключение

В результате использования кода Python процесс подготовки исходных данных для прогнозного анализа был автоматизирован и существенно упростился. Таким образом, удалось реализовать функции обработки и управления данными на языке Python.

Список использованных источников

1. Губин Е.И. Использование программных инструментов SAS для подготовки больших данных. Современные технологии, экономика и образование: Сборник трудов Всероссийской научно-методической конференции, Томск, 2-4 сентября 2020 Томск: ТПУ- С. 53-55
2. Губин Е.И. Методика подготовки больших данных для прогнозного анализа «Наука и бизнес: пути развития». Выпуск №3(105). 2020, 2020. – [С. 33-35].