Добрый день, чтобы запустить программу необходимо сохранить датасет ‘TR_nef.xlxs’, как ‘TR_nef.csv’. Используется библиотека ‘pandas’ для работы с большими массивами информации.

Итак, на входе имеется выборка значений с датчиков на скважинах, выберем необходимые столбцы, конвертируем дату в ‘YYYY-MM’ для удобства отображения графиков, с использованием кодировки 'cp1251' для российских литер.

```
# load data from CSV file
# Using encoding for russian letters
# String type of column with names of wells
# Convert data to "YYYY-MM" format

df = pd.read_csv('TR_nef.csv', encoding = 'cp1251', engine='python', sep = ';', decimal = ',', thousands = ' ', header = 4,
            dtype = {'Скважина':str}, usecols = [0,1,6,18,19,21,22,23,27,59],
            names = 'Дата;Скважина;Диаметр экспл.колонны, мм;Буферное давление, атм;\
            Давление в линии, атм;Динамическая высота, м;Затрубное давление, атм;\
            Давление на приеме, атм;Обводненность, %;Глубина спуска, м'.split(';'),
            converters = {'Дата': lambda x: str('{0:02.2f}'.format((float(x) - int(float(x))) * 10000 + int(float(x)) / 100))})

# copy main dataset
dff = df.copy()

# put the required column "Давление на приеме" to the end of the dataframe
reindex = pd.DataFrame(df.pop('Давление на приеме, атм'))
df = df.join(reindex)

# delete all rows with NaN (empty) values
df = df.dropna()
```

Тепловые карты

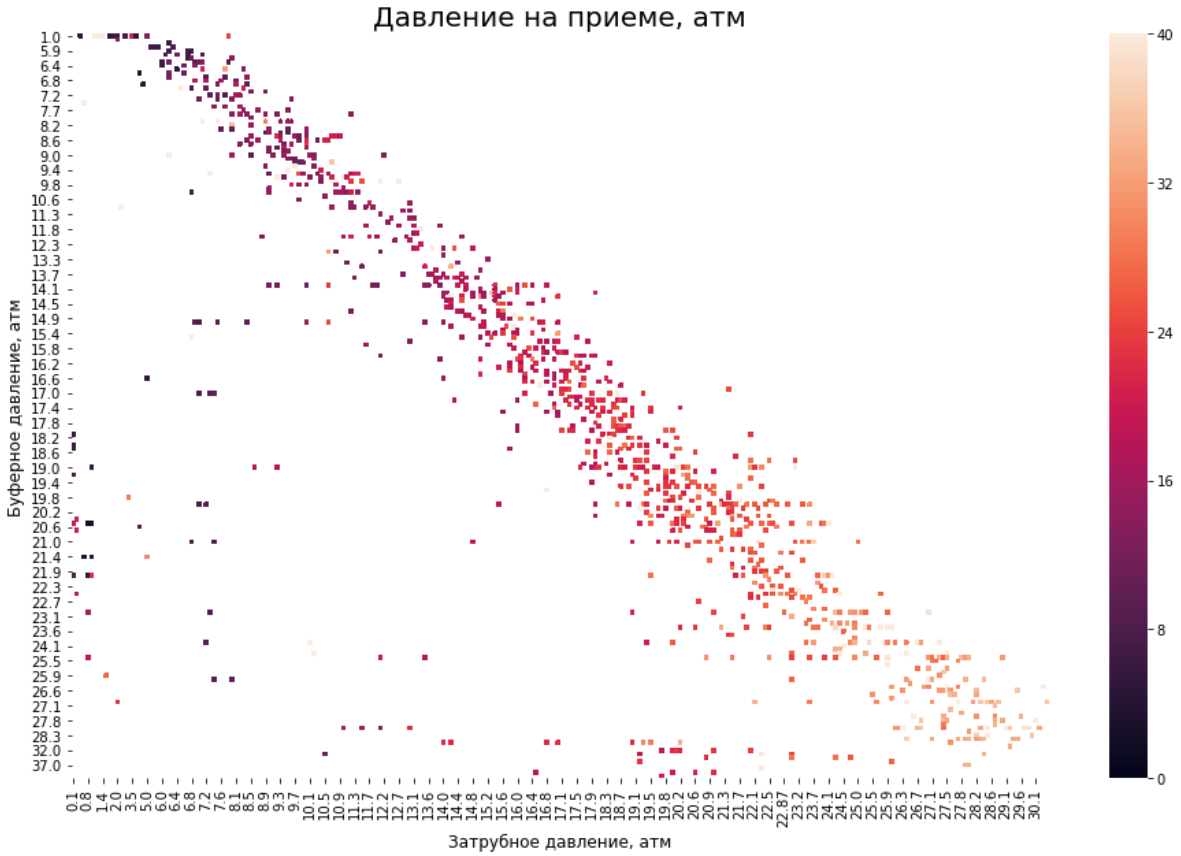
Для реализации отображения тепловых карт использована библиотека ‘seaborn’. Хитмап строится по трём спискам значений, показатели которых не должны дублироваться в первых двух колонках. Удаляются дубликаты, создается пивот-таблица, строится тепловая карта.

```
# delete duplicates in rows in required columns ('Давление в линии, атм', 'Затрубное давление, атм')
df1 = df.drop_duplicates(subset=['Давление в линии, атм', 'Затрубное давление, атм'], keep=False)

# creating pivot table
pivot_table1 = df1.pivot('Давление в линии, атм', 'Затрубное давление, атм', 'Давление на приеме, атм')

# creating plot of the heatmap
plt.figure(figsize=(16, 10))
plt.xlabel('', size = 12)
plt.ylabel('', size = 12)
plt.title('Давление на приеме, атм', size = 20)
sns.heatmap(pivot_table1, vmin = 0, vmax = 40, square = True)
plt.show()
```

Как известно, искомые значения в колонке ‘давление на приеме’, поэтому тепловые карты строим для них по осям, выбранных из заданных 7 параметров. Было выбрано 3 наиболее информативных тепловых карты из всех возможных вариантов. По следующим значениям:

- давление в линии, затрубное давление, давление на приеме;
- буферное давление, затрубное давление, давление на приеме;
- буферное давление, давление в линии, давление на приеме.


Давление на приеме, атм
Давление в линии, атм
Затрубное давление, атм

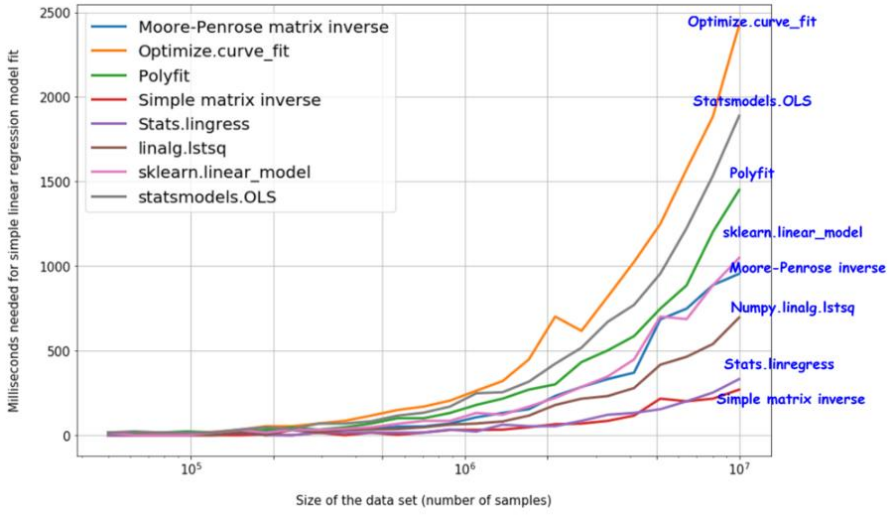

Давление на приеме, атм
Буферное давление, атм
Затрубное давление, атм

Из графиков можно сделать следующие выводы:

- при высоком давлении в линии и большом затрубном давлении, значения давления на приеме систематически выше, чем при малом давлении в линии и малом затрубном давлении;
- при большем значении буферного давления и затрубном давлении, значения давления на приеме выше, чем при малых значениях параметров;
- та же зависимость при анализе буферного давления и давления в линии;
- графики искомого давления на приеме характеризуются линейным возрастании величины, поэтому во второй части для предсказания используется модель ‘линейной регрессии’.

Классическое обучение с учителем

Для предсказания количественного параметра, при работе с массивом с известными данными, использован метод ‘обучения с учителем’ – линейную регрессию.



Разделяем исходную таблицу на массив предикторов и массив зависимой переменной. Используется ‘sklearn.linear_model’ библиотека из-за относительной простоты использования и скорости работы.

На выходе отметим значение среднеквадратической ошибки.

```
# x - a table with factors, exept date and wells' names columns ('Диаметр экспл.колонны, буферное давление и тд')
# y - a table with a dependent variable ('Давление на приеме')
x = df.iloc[:,2:df.shape[1]-1]
y = df.iloc[:,-1]
# use library sklearn.linear_model for application linear regression
# create an empty model of linear regression
skm = lm.LinearRegression()
# calculate parameters
skm.fit(x, y)
# show them
skm.intercept_, skm.coef_
# prediction
skm.predict(x)
# calculation mean squared error
mse = np.mean((y - skm.predict(x)) ** 2)
print(mse)
```

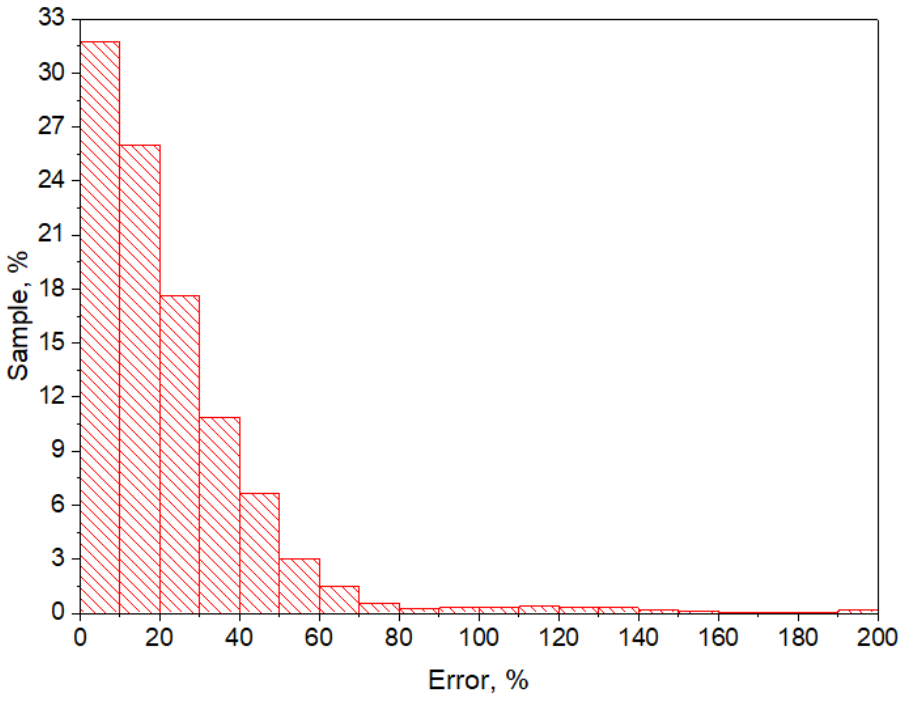
```
83.19958513516593
```

Выведены в таблицу данные зависимой переменной, предсказанные значения и разницу.

```
# creating an info table with parameters (coefficients) of linear regression
pd.DataFrame(list(zip(x.columns, skm.coef_)), columns=['features','est.coef.'])
# making output table of source and predicted values
result = pd.DataFrame(list(zip(y, skm.predict(x), y - skm.predict(x))), columns=['y', 'yhat', 'difference'])
# write a csv file to draw nice histogram in Origin
result.to_csv('result.csv', index = False, sep=',', encoding='utf-8')
```

Гистограмма ошибки

Построим гистограмму ошибки в осях: доля примеров из тестовой выборки, %, величина ошибки, %.



Около трети выборки меньше 10% ошибки, практически вся выборка удовлетворяет условию – меньше 50% ошибки.

Дополнительный вариант выборки значений для линейной регрессии

Изначально идея состояла в разделении исходного массива на 2 части: с известными значениями давления на приеме и пустыми.

Создавалось и заполнялось 2 датафрейма, брался датафрейм с известными величинами зависимой переменной, так же делился на 2 части: предикторы и искомые. Пробелы в массиве предикторов заполнялись средними значениями по столбцам, из-за чего результат среднеквадратической ошибки хуже. Гистограмма также отображает больший разброс ошибки

```
#labels - written handly
#creating predictors matrix (df1 pd.dataframe) and result matrix (df0 pd.dataframe)

labels = 'Дата;Скважина;Диаметр экспл.колонны, мм;Буферное давление, атм;Давление в линии, атм;\
Динамическая высота, м;Затрубное давление, атм;Обводненность, %;Глубина спуска, м;Давление на приеме, атм'
df0 = pd.read_csv(StringIO(labels),sep = ';')
df1 = pd.read_csv(StringIO(labels),sep = ';')
new_line_df0 = df0.shape[1]
new_line_df1 = df1.shape[1]

#filling dataframes, using copy of main dataset - dataframe 'dff'

for index, row in dff.iterrows():
    if pd.isnull(row['Давление на приеме, атм']):
        new_line_df0 = new_line_df0 + 1
        df0.loc[new_line_df0] = dff.iloc[index]
    else:
        new_line_df1 = new_line_df1 + 1
        df1.loc[new_line_df1] = dff.iloc[index]
```

```
#change nodata to mean value
df1 = df1.fillna(df1.mean())
# x - a table with factors x1,x2,x3
x1 = df1.iloc[:,2:df1.shape[1]-1]
# y - a table with a dependent variable
y1 = df1.iloc[:,-1]
# create an empty model
skm1 = lm.LinearRegression()
# calculate parameters
skm1.fit(x1, y1)
# show them
skm1.intercept_, skm1.coef_
# prediction
skm1.predict(x1)
# calculation mean squared error
mse1 = np.mean((y1 - skm1.predict(x1)) ** 2)
print(mse1)
```

```
172.6035715301799
```