ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

по курсу

«Data Science»

На тему: «Прогнозирование конечных свойств новых материалов (композиционных материалов)»

Слушатель: Виталий Благодатских

Москва, 2022

Постановка задачи

Цель

Спрогнозировать ряд конечных свойств получаемых композиционных материалов

1. **Построить модели прогнозирования характеристик «модуль упругости при растяжении», «прочность при растяжении».**
2. **Построить нейронную сеть дающую рекомендации параметра «соотношение матрица-наполнитель».**

Исходные данные:

Два датасета, содержащие данные о начальных свойствах компонентов композиционных материалов (количество связующего, наполнителя, температурный режим отверждения и т.д.).

Актуальность

Композитные материалы широко применяются в современной технике. Они позволяют сочетать лучшие качества составляющих их материалов, формируя лёгкие, прочные, износоустойчивые изделия. Свойства и назначение композиционных материалов определяются их строением, составом и структурой. Использование волокнистых материалов позволяет создать высокопрочные композиты, эластичные полимеры (например, полиуретан) повышают износоустойчивость, введение абразивных материалов позволяет получить композиты устойчивые к истиранию. Создание новых композитных материалов для специального применения имеет важное значение.

Созданные прогнозные модели помогут сократить количество проводимых испытаний, а также пополнить базу данных материалов возможными новыми характеристиками материалов, и цифровыми двойниками новых композитов.

Разведочный анализ данных и визуализация исходных данных

Формирование датасета

```
df.head()
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	н
0.0	1.857143	2030.0	738.736842	30.00	22.267857	100.000000	210.0	70.0	3000.0	220.0	0.0	
1.0	1.857143	2030.0	738.736842	50.00	23.750000	284.615385	210.0	70.0	3000.0	220.0	0.0	
2.0	1.857143	2030.0	738.736842	49.90	33.000000	284.615385	210.0	70.0	3000.0	220.0	0.0	
3.0	1.857143	2030.0	738.736842	129.00	21.250000	300.000000	210.0	70.0	3000.0	220.0	0.0	
4.0	2.771331	2030.0	753.000000	111.86	22.267857	284.615385	210.0	70.0	3000.0	220.0	0.0	

Описательная статистика

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Соотношение матрица-наполнитель	1023.0	2.930366	0.913222	0.389403	2.317887	2.906878	3.552660	5.591742
Плотность, кг/м3	1023.0	1975.734888	73.729231	1731.764635	1924.155467	1977.621657	2021.374375	2207.773481
модуль упругости, ГПа	1023.0	739.923233	330.231581	2.436909	500.047452	739.664328	961.812526	1911.536477
Количество отвердителя, м.%	1023.0	110.570769	28.295911	17.740275	92.443497	110.564840	129.730366	198.953207
Содержание эпоксидных групп,%_2	1023.0	22.244390	2.406301	14.254985	20.608034	22.230744	23.961934	33.000000
Температура вспышки, С_2	1023.0	285.882151	40.943260	100.000000	259.066528	285.896812	313.002106	413.273418
Поверхностная плотность, г/м2	1023.0	482.731833	281.314690	0.603740	266.816645	451.864365	693.225017	1399.542362
Модуль упругости при растяжении, ГПа	1023.0	73.328571	3.118983	64.054061	71.245018	73.268805	75.356612	82.682051
Прочность при растяжении, МПа	1023.0	2466.922843	485.628006	1036.856605	2135.850448	2459.524526	2767.193119	3848.436732
Потребление смолы, г/м2	1023.0	218.423144	59.735931	33.803026	179.627520	219.198882	257.481724	414.590628
Угол нашивки, град	1023.0	44.252199	45.015793	0.000000	0.000000	0.000000	90.000000	90.000000
Шаг нашивки	1023.0	6.899222	2.563467	0.000000	5.080033	6.916144	8.586293	14.440522
Плотность нашивки	1023.0	57.153929	12.350969	0.000000	49.799212	57.341920	64.944961	103.988901

Проверка на пропуски и дубликаты

```
df.isna().sum()
```

```
Соотношение матрица-наполнитель          0
Плотность, кг/м3                         0
модуль упругости, ГПа                    0
Количество отвердителя, м.%              0
Содержание эпоксидных групп,%_2          0
Температура вспышки, С_2                 0
Поверхностная плотность, г/м2            0
Модуль упругости при растяжении, ГПа     0
Прочность при растяжении, МПа            0
Потребление смолы, г/м2                  0
Угол нашивки, град                       0
Шаг нашивки                              0
Плотность нашивки                        0
dtype: int64
```

Пропусков нет

```
df.duplicated().sum()
```

```
0
```

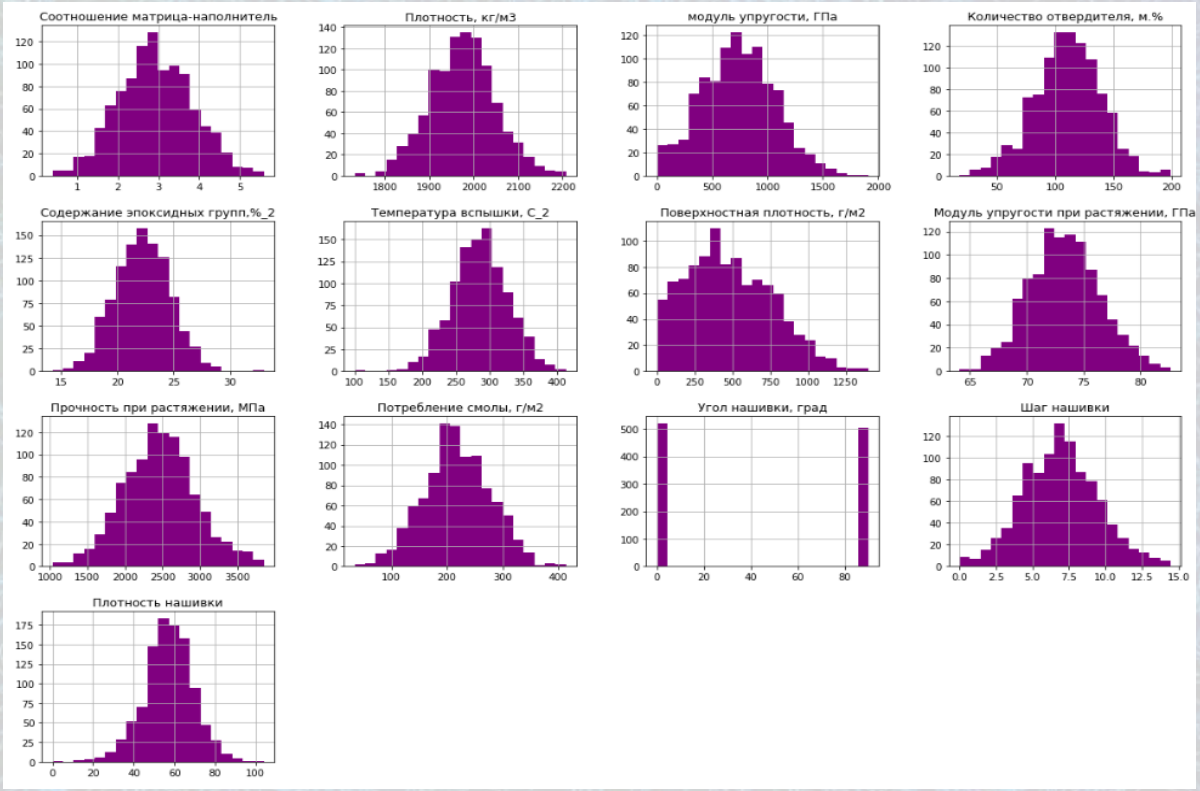
Дубликатов нет

Преобразование в бинарный вид

```
# Преобразуем значения "Угол нашивки" с помощью LabelEncoder
le = preprocessing.LabelEncoder()
df['Угол нашивки, град'] = le.fit_transform(df['Угол нашивки, град'].values)
df['Угол нашивки, град']
```

Разведочный анализ данных и визуализация исходных данных

Гистограммы распределения переменных

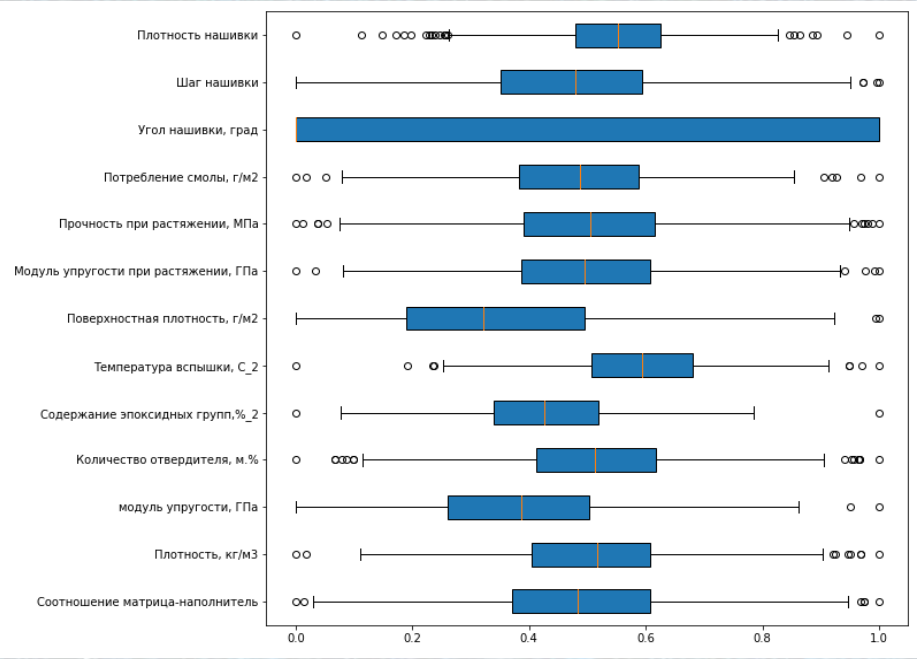


Из всех параметров, помимо «Угла нашивки», имеющего всего два значения, выделяются «Поверхностная плотность, г/м2» и «модуль упругости, ГПа» форма распределения менее других походит на нормальное

Разведочный анализ данных и визуализация исходных данных

«Ящики с усами»

До удаления выбросов

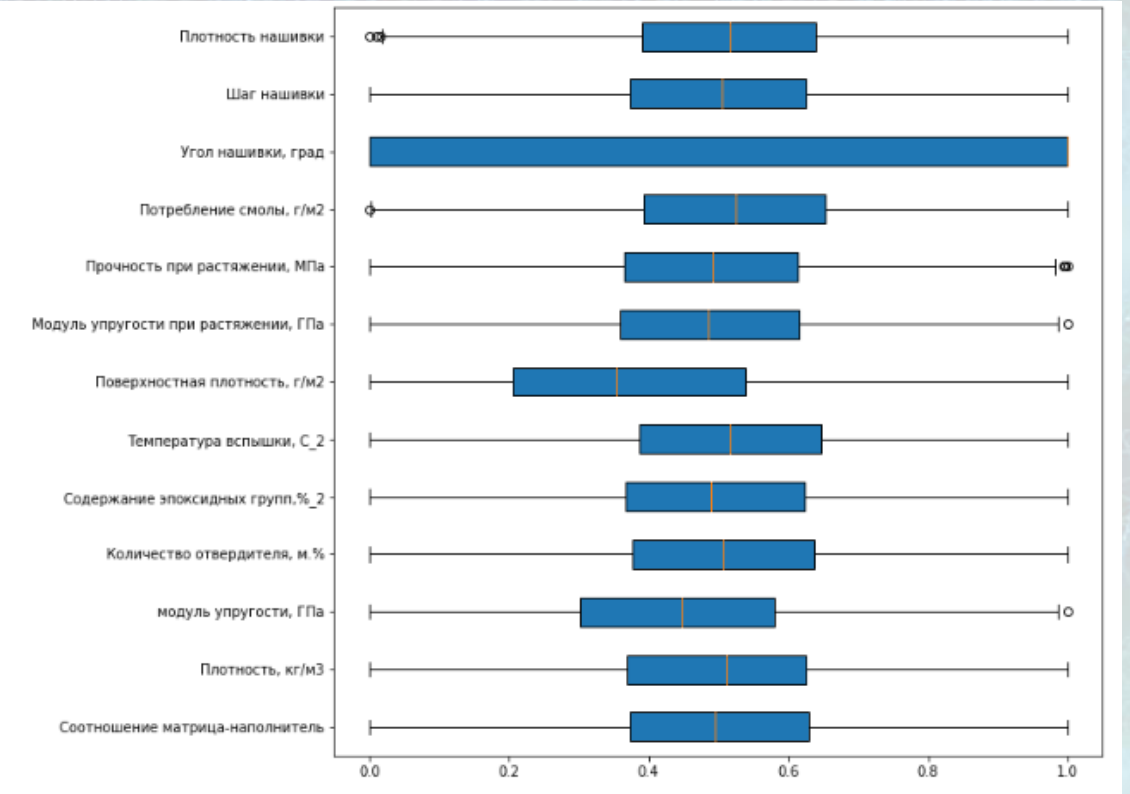



```
for x in df.columns:
    q75, q25 = np.percentile(df.loc[:,x],[75,25])
    qr = q75-q25


    max = q75+(1.5*qr)
    min = q25-(1.5*qr)

    df.loc[df[x] < min,x] = np.nan
    df.loc[df[x] > max,x] = np.nan
```

```
# Смотрим сколько выбросов получилось по каждому столбцу
df.isnull().sum()
```

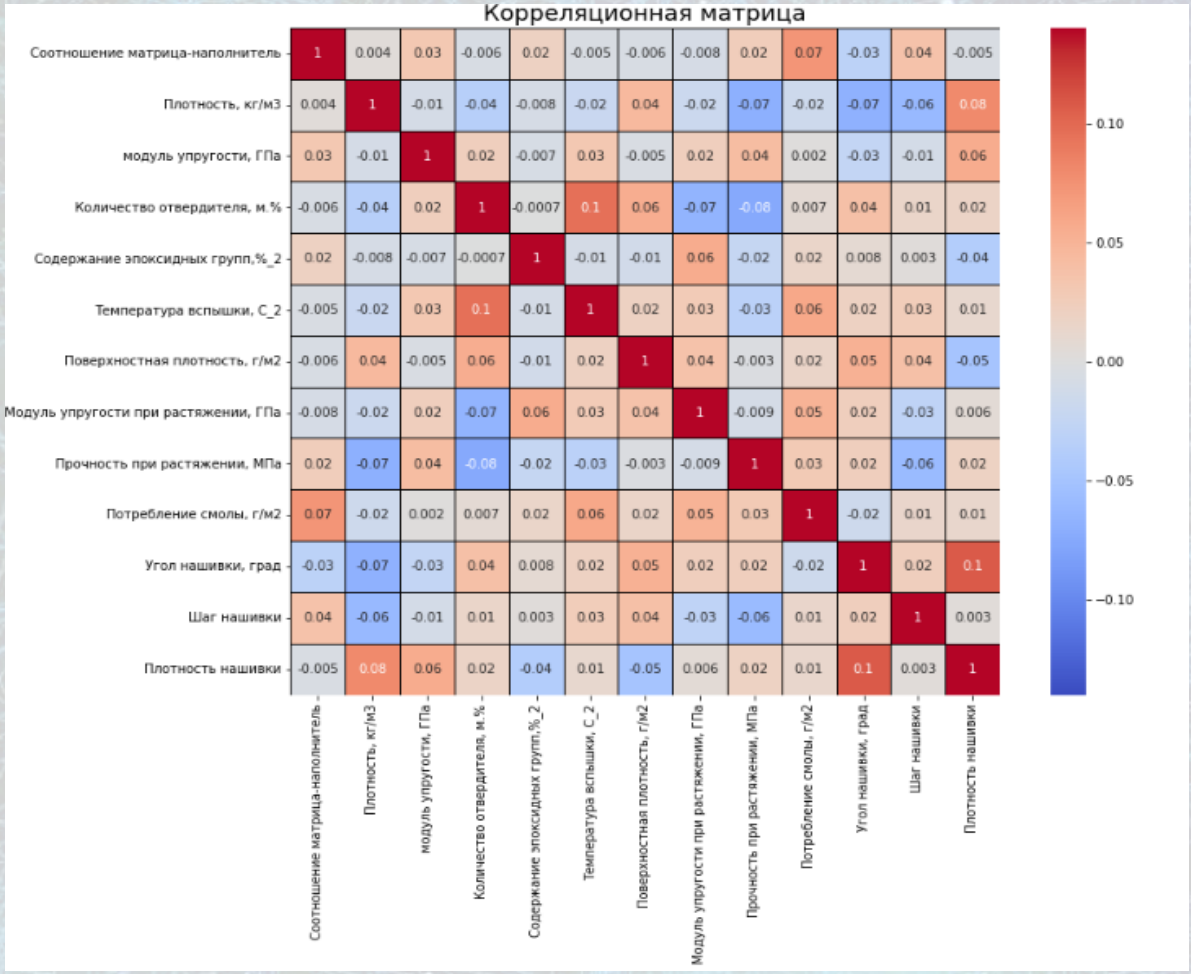
```
# Удаляем выбросы
df = df.dropna(axis = 0)
```

После удаления выбросов



Разведочный анализ данных и визуализация исходных данных

Матрица кореляции



Предобработка данных

Выполняем нормализацию

```
# Используем MinMaxScaler()
min_max_scaler = preprocessing.MinMaxScaler()
df_norm = pd.DataFrame(min_max_scaler.fit_transform(df), columns=df.columns)
df_norm
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	н
0	0.274768	0.651097	0.447061	0.079153	0.607435	0.509164	0.162230	0.280303	0.712590	0.529221	0.0	(
1	0.274768	0.651097	0.447061	0.630983	0.418887	0.583596	0.162230	0.280303	0.712590	0.529221	0.0	(
2	0.466552	0.651097	0.455721	0.511257	0.495653	0.509164	0.162230	0.280303	0.712590	0.529221	0.0	(
3	0.465836	0.571539	0.452685	0.511257	0.495653	0.509164	0.162230	0.280303	0.712590	0.529221	0.0	(
4	0.424236	0.332865	0.468508	0.511257	0.495653	0.509164	0.162230	0.280303	0.712590	0.529221	0.0	(
...	...	...	...	...	...	...	...	...	...	...	...	
931	0.361662	0.444480	0.552781	0.337550	0.333908	0.703458	0.161609	0.475147	0.463043	0.207613	1.0	(
932	0.607674	0.704373	0.268550	0.749605	0.294428	0.362087	0.271207	0.464422	0.452087	0.182974	1.0	(
933	0.573391	0.498274	0.251612	0.501991	0.623085	0.334063	0.572959	0.578740	0.575296	0.585446	1.0	(
934	0.662497	0.748688	0.448724	0.717585	0.267818	0.466417	0.496511	0.535142	0.334513	0.451779	1.0	(
935	0.684036	0.280923	0.251903	0.632264	0.888354	0.588206	0.587373	0.551972	0.654075	0.443749	1.0	(

936 rows × 13 columns

Выполняем стандартизацию

```
# Используем StandardScaler
df_standart = preprocessing.StandardScaler().fit(df_norm)
df_standart = df_standart.transform(df_norm)
df_standart = pd.DataFrame(df_standart, columns=df.columns)
df_standart
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Модуль упругости при растяжении, ГПа	Прочность при растяжении, МПа	Потребление смолы, г/м2	Угол нашивки, град	н
0	-1.196260	0.790727	0.001489	-2.254199	0.643790	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787	-
1	-1.196260	0.790727	0.001489	0.669189	-0.400666	0.354488	-0.974837	-1.088732	1.148666	0.041294	-1.023787	-
2	-0.172802	0.790727	0.044904	0.034925	0.024577	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787	-
3	-0.176623	0.366820	0.029685	0.034925	0.024577	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787	-
4	-0.398622	-0.904900	0.209271	0.034925	0.024577	-0.036187	-0.974837	-1.088732	1.148666	0.041294	-1.023787	-
...	...	...	...	...	...	...	...	...	...	...	...	
931	-0.732548	-0.310188	0.531478	-0.885307	-0.871403	0.983609	-0.977698	-0.070548	-0.172989	-1.602276	0.976766	
932	0.580295	1.074592	-0.893411	1.297605	-1.090103	-0.808159	-0.472549	-0.126591	-0.231014	-1.728193	0.976766	
933	0.397344	-0.023557	-0.978322	-0.014163	0.730480	-0.955250	0.918256	0.470794	0.421529	0.328626	0.976766	-
934	0.872860	1.310716	0.009825	1.127974	-1.237507	-0.260555	0.565897	0.242965	-0.853711	-0.354474	0.976766	-
935	0.987800	-1.181662	-0.976862	0.675976	2.199930	0.378680	0.984693	0.330915	0.838757	-0.395510	0.976766	-

936 rows × 13 columns

Разработка моделей

Линейная регрессия

```
mu = df_standart.drop(['Модуль упругости при растяжении, ГПа','Прочность при растяжении, МПа'], axis = 1)
pr = df_standart.drop(['Модуль упругости при растяжении, ГПа','Прочность при растяжении, МПа'], axis = 1)
mu
```

	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2	Поверхностная плотность, г/м2	Потребление смолы, г/м2	Угол нашивки, град	Шаг нашивки	Плотность нашивки
0	-1.196260	0.790727	0.001489	-2.254199	0.643790	-0.036187	-0.974837	0.041294	-1.023787	-1.162360	0.226834
1	-1.196260	0.790727	0.001489	0.669189	-0.400666	0.354488	-0.974837	0.041294	-1.023787	-0.763689	-0.930436
2	-0.172802	0.790727	0.044904	0.034925	0.024577	-0.036187	-0.974837	0.041294	-1.023787	-0.763689	-0.040228
3	-0.176623	0.366820	0.029685	0.034925	0.024577	-0.036187	-0.974837	0.041294	-1.023787	-0.763689	0.226834
4	-0.398622	-0.904900	0.209271	0.034925	0.024577	-0.036187	-0.974837	0.041294	-1.023787	-0.763689	1.117043
...	...	...	...	...	...	...	...	...	...	...	...
931	-0.732548	-0.310188	0.531478	-0.885307	-0.871403	0.983609	-0.977698	-1.602276	0.976766	0.861447	-0.928676
932	0.580295	1.074592	-0.893411	1.297605	-1.090103	-0.808159	-0.472549	-1.728193	0.976766	1.455162	-0.329475
933	0.397344	-0.023557	-0.978322	-0.014163	0.730480	-0.955250	0.918256	0.328626	0.976766	-1.098113	0.906035
934	0.872860	1.310716	0.009825	1.127974	-1.237507	-0.260555	0.565897	-0.354474	0.976766	-0.240153	0.072034
935	0.987800	-1.181662	-0.976862	0.675976	2.199930	0.378680	0.984693	-0.395510	0.976766	-0.333562	1.778865

936 rows × 11 columns

```
mu_X = mu
mu_Y = df_standart['Модуль упругости при растяжении, ГПа']
pr_X = pr
pr_Y = df_standart['Прочность при растяжении, МПа']
```

```
# Разобьем датасет mu_X и датасет pr_X на тестовую и тренировочную выборки.

# mu делим на тестовую и тренировочную выборки, зависимая  mu_Y - Модуль упругости при растяжении#
mu_X_train, mu_X_test, mu_Y_train,  mu_Y_test = train_test_split(mu_X, mu_Y, test_size = 0.30, random_state=1)
```

Пишем функцию, которая рассчитывает среденнее значение по тестовой выборке. Со средним будем сравнивать результаты предсказаний моделей

```
def mean_model(mu_Y_test):
    return [np.mean(mu_Y_test) for _ in range(len(mu_Y_test))]
mu_Y_pred_mean = mean_model(mu_Y_test)
mean_absolute_error (mu_Y_test, mu_Y_pred_mean)
```

```
0.8289382478364875
```

```
def mean_model(pr_Y_test):
    return [np.mean(pr_Y_test) for _ in range(len(pr_Y_test))]
pr_Y_pred_mean = mean_model(pr_Y_test)
mean_absolute_error (pr_Y_test, pr_Y_pred_mean)
```

```
0.7880988878953844
```

Для модуля упругости при растяжении

```
lin_reg= LinearRegression()
lin_reg.fit(mu_X_train, mu_Y_train)
```

```
LinearRegression()
```

```
# Предсказание значения для mu
mu_Y_pred = lin_reg.predict(mu_X_test).round(3)
mu_lin = pd.DataFrame({'Actual': mu_Y_test, 'Predicted': mu_Y_pred})
mu_lin.head()
```

	Actual	Predicted
386	-1.384355	0.109
41	0.873654	-0.026
725	-0.528017	0.061
605	0.504928	0.130
35	-0.601519	-0.072

```
# Результаты модели
mu_mse_lin_elast = mean_squared_error(mu_Y_test, mu_Y_pred)
print("MAE: ", mean_absolute_error(mu_Y_test, mu_Y_pred))
print("MSE: ", mu_mse_lin_elast)
print("RMSE: ", np.sqrt(mu_mse_lin_elast))
```

```
MAE:  0.8385264893496368
MSE:  1.0347000196871035
RMSE:  1.0172020545039728
```

Разработка моделей

Метод К-ближайших соседей (KNeighborsRegressor)

Для прочности при растяжении

```
GSCV_kn_pr = GridSearchCV(kn, kn_params, n_jobs=-1, cv=10)
GSCV_kn_pr.fit(pr_X_train, pr_Y_train)
GSCV_kn_pr.best_params_
```

```
{'algorithm': 'brute', 'n_neighbors': 235, 'weights': 'distance'}
```

```
kn_pr = GSCV_kn_pr.best_estimator_
print(f'R2-score KNR для прочности при растяжении: {kn_pr.score(pr_X_test, pr_Y_test).round(3)}')
```

```
R2-score KNR для прочности при растяжении: -0.009
```

```
kn_pr_result = pd.DataFrame({
   'Model': 'KNeighborsRegressor_pr',
   'MAE': mean_absolute_error(pr_Y_test, kn_pr.predict(pr_X_test)),
   'R2 score': kn_pr.score(pr_X_test, pr_Y_test).round(3)
}, index=['Прочность при растяжении'])

models4 = models3.append(kn_pr_result)
models4
```

	Model	MAE	R2 score
Модуль упругости при растяжении	LinearRegression_mu	0.838530	-0.019
Прочность при растяжении	LinearRegression_pr	0.797591	-0.018
Модуль упругости при растяжении	KNeighborsRegressor_mu	0.840831	-0.015
Прочность при растяжении	KNeighborsRegressor_pr	0.792277	-0.009

Разработка моделей

Модель «Случайного леса» для модуля упругости при растяжении

```
{'n_estimators': 220,
 'min_samples_split': 40,
 'min_samples_leaf': 5,
 'max_depth': 1,
 'criterion': 'absolute_error',
 'bootstrap': 'True'}
```

```
# Создаем сетку параметров на основе случайного поиска
# Создаем модель поиска по сетке с перекрестной проверкой, количество блоков равно 10
rf = RandomForestRegressor()
rf_param = {
    'n_estimators' : range(10, 1000, 10),
    'criterion' : ['squared_error', 'absolute_error', 'poisson'],
    'max_depth' : range(1, 7),
    'min_samples_split' : range(20, 50, 5),
    'min_samples_leaf' : range(2, 8),
    'bootstrap' : ['True', 'False']
}
mu_rf = RandomizedSearchCV(rf, rf_param, n_jobs=-1, cv=10, verbose=4)
# Обучаем модель
mu_rf.fit(mu_X_train, mu_Y_train)
# Ищем лучшие параметры для модели
mu_rf.best_params_
```

```
Fitting 10 folds for each of 10 candidates, totalling 100 fits
```

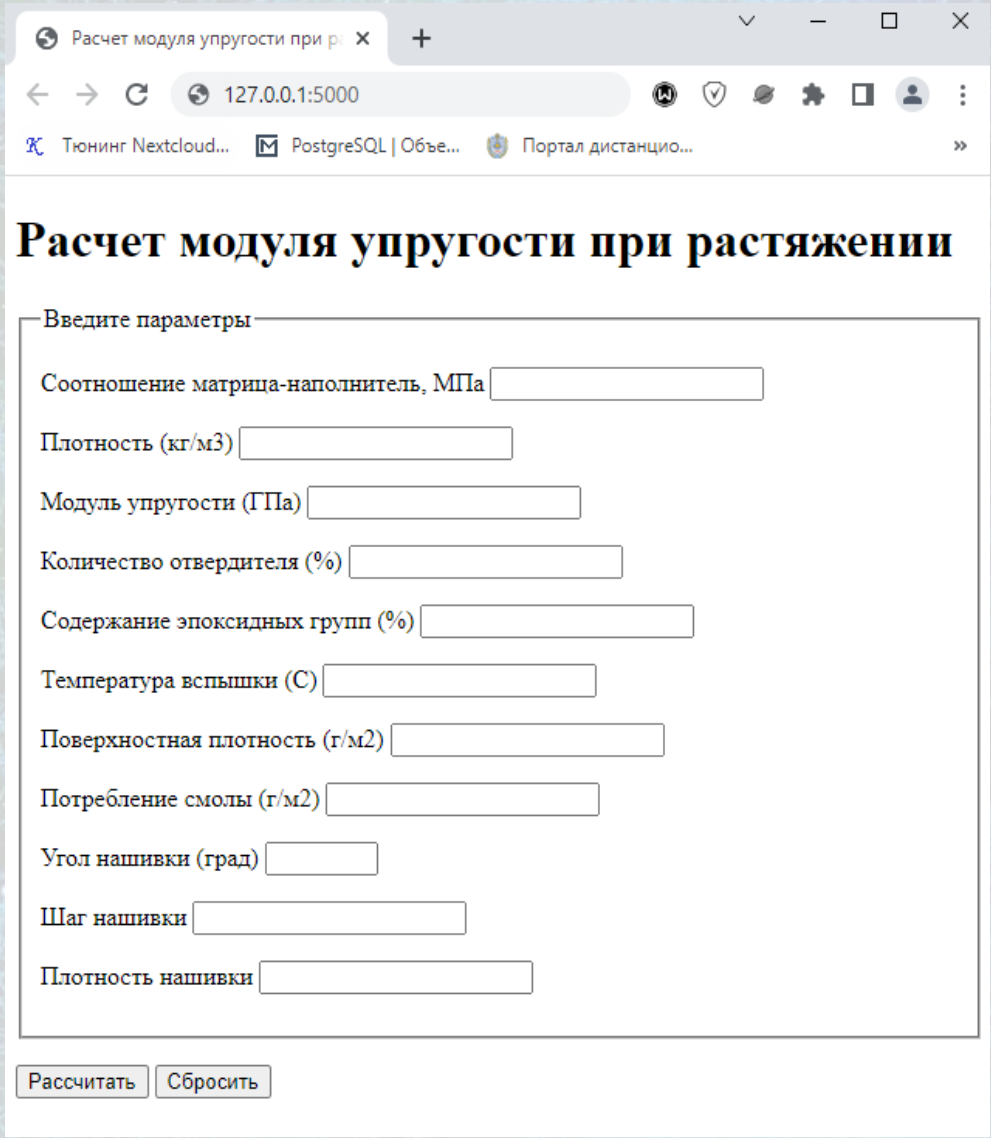
	Model	MAE	R2 score
Модуль упругости при растяжении	LinearRegression_mu	0.838530	-0.019
Прочность при растяжении	LinearRegression_pr	0.797591	-0.018
Модуль упругости при растяжении	KNeighborsRegressor_mu	0.840831	-0.015
Прочность при растяжении	KNeighborsRegressor_pr	0.792277	-0.009
Модуль упругости при растяжении	Random Forest Regressor_mu	0.832147	-0.006
Прочность при растяжении	Random Forest Regressor_pr	0.791913	-0.004

```
#Предсказываем значения
mu_rf_pred = mu_rf.predict(mu_X_test).round(3)

mu_grid_rf = mu_rf.best_estimator_
print(f'R2-score RFRegr для модуля упругости при растяжении: {mu_grid_rf.score(mu_X_test, mu_Y_test).round(3)}')
```

```
R2-score RFRegr для модуля упругости при растяжении: -0.006
```

```
mu_rf_result = pd.DataFrame({
   'Model': 'Random Forest Regressor_mu',
   'MAE': mean_absolute_error(mu_Y_test, mu_grid_rf.predict(mu_X_test)),
   'R2 score': mu_grid_rf.score(mu_X_test, mu_Y_test).round(3)
}, index=['Модуль упругости при растяжении'])

models5 = models4.append(mu_rf_result)
models5
```

Разработка web-приложения на платформе Flask



Спасибо за внимание