ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Прогнозирование конечных свойств новых материалов (композиционных материалов)

ФИО докладчика: Мукатова А.К.

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Разведочный анализ данных

```
#Чтобы проверить наличие дубликатов в данных, мы можем использовать функцию duplicated
full_df.duplicated().sum()
```

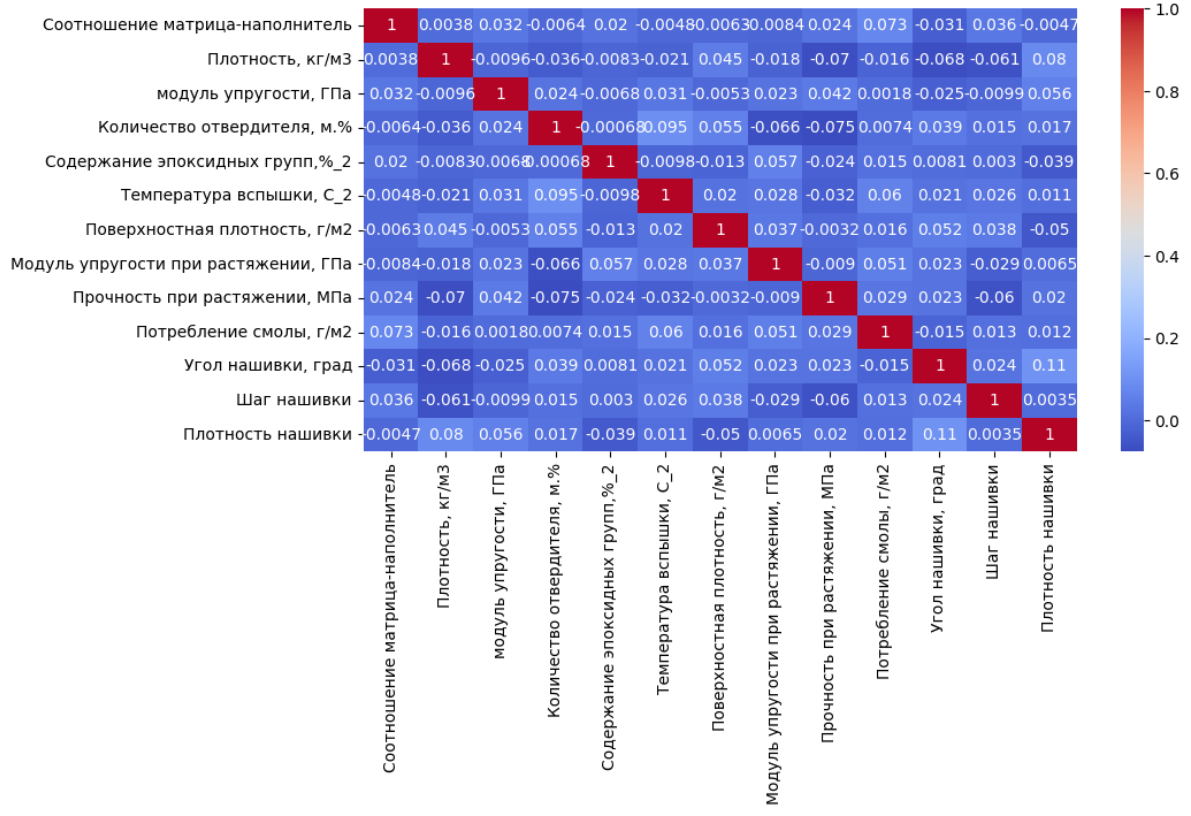
```
0
```

```
#Чтобы проверить наличие пропущенных значений в данных, мы можем использовать функцию isnull
full_df.isnull().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
```

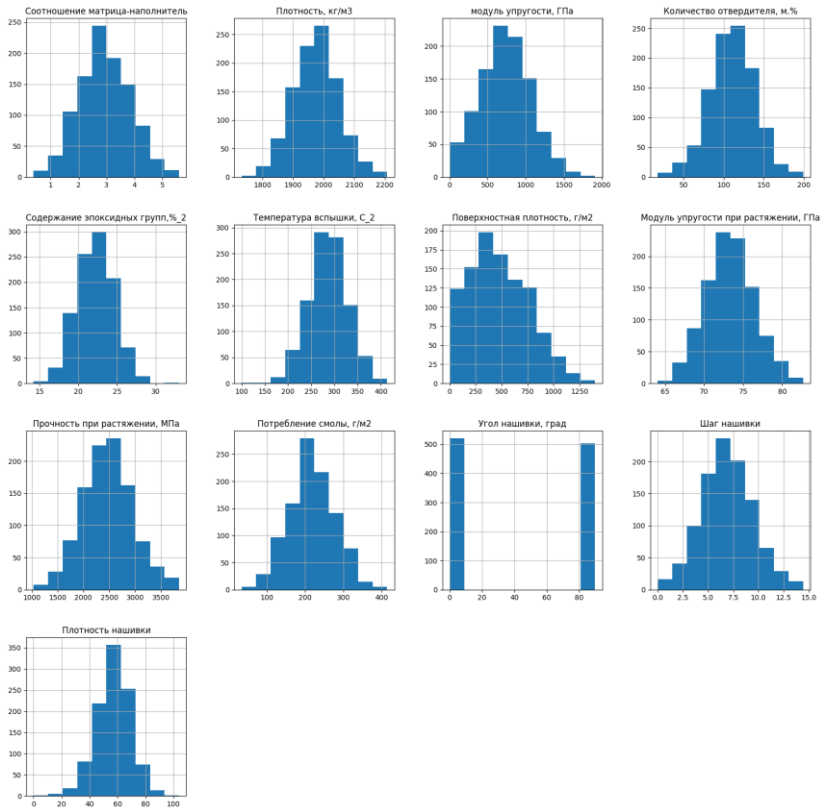
```
#Чтобы посмотреть на корреляцию между столбцами, мы можем использовать функцию corr
full_df.corr()
```

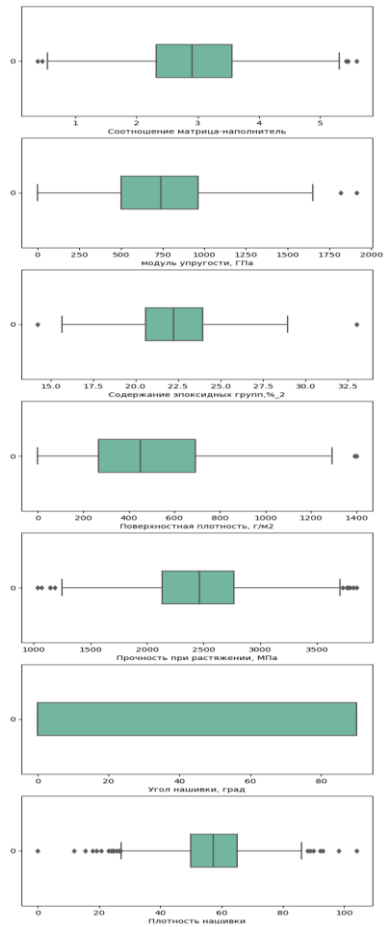
	Соотношение матрица-наполнитель	Плотность, кг/м3	модуль упругости, ГПа	Количество отвердителя, м.%	Содержание эпоксидных групп,%_2	Температура вспышки, С_2
Соотношение матрица-наполнитель	1.000000	0.003841	0.031700	-0.006445	0.019766	-0.004776
Плотность, кг/м3	0.003841	1.000000	-0.009647	-0.035911	-0.008278	-0.020695
модуль упругости, ГПа	0.031700	-0.009647	1.000000	0.024049	-0.006804	0.031174
Количество отвердителя, м.%	-0.006445	-0.035911	0.024049	1.000000	-0.000684	0.095193

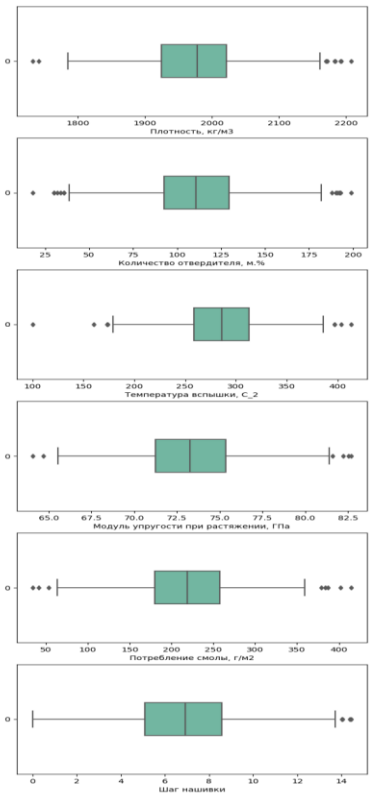


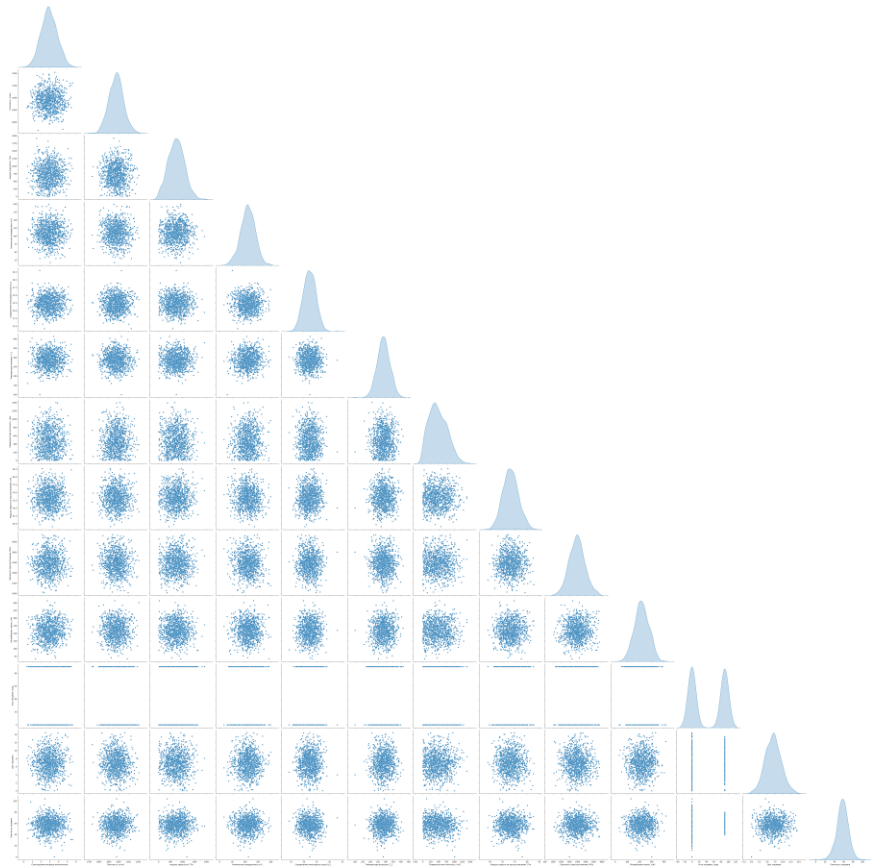
ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Визуализация









ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Разработка и обучение моделей

```
#Обучаем модель на тренировочных данных
dectree_reg.fit(X_train_dex, y_train_dex)
#Выводим метрики качества модели на тестовых данных
print(mean_absolute_error(y_test_dex, dectree_reg.predict(X_test_dex)))
print(mean_squared_error(y_test_dex, dectree_reg.predict(X_test_dex)))
print(sqrt(mean_squared_error(y_test_dex, dectree_reg.predict(X_test_dex))))
print(r2_score(y_test_dex, dectree_reg.predict(X_test_dex)))
```

```
2.3879803050537354
8.757655738206834
2.9593336645614725
-0.0049838343000516705
```

```
print (X_train_dex.shape)
print (X_test_dex.shape)
print (y_train_dex.shape)
print (y_test_dex.shape)
```

```
(655, 12)
(281, 12)
(655, 1)
(281, 1)
```

```
#Обучаем модель на тренировочных данных
lin_reg.fit(X_train_dex, y_train_dex)
#Выводим метрики качества модели на тестовых данных
print(mean_absolute_error(y_test_dex, lin_reg.predict(X_test_dex)))
print(mean_squared_error(y_test_dex, lin_reg.predict(X_test_dex)))
print(sqrt(mean_squared_error(y_test_dex, lin_reg.predict(X_test_dex))))
print(r2_score(y_test_dex, lin_reg.predict(X_test_dex)))
```

```
2.4145733303816472
8.759731042865322
2.959684280943716
-0.0052219856610293824
```

```
# Обучаем модель на тренировочных данных
knr.fit(X_train_dex, y_train_dex)
# Выводим метрики качества модели на тестовых данных
print(mean_absolute_error(y_test_dex, knr.predict(X_test_dex)))
print(mean_squared_error(y_test_dex, knr.predict(X_test_dex)))
print(sqrt(mean_squared_error(y_test_dex, knr.predict(X_test_dex))))
print(r2_score(y_test_dex, knr.predict(X_test_dex)))
```

```
2.4399305107070863
8.969518116258147
2.994915377144761
-0.029296078512740076
```

ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Нейронные сети для соотношения матрица-наполнитель

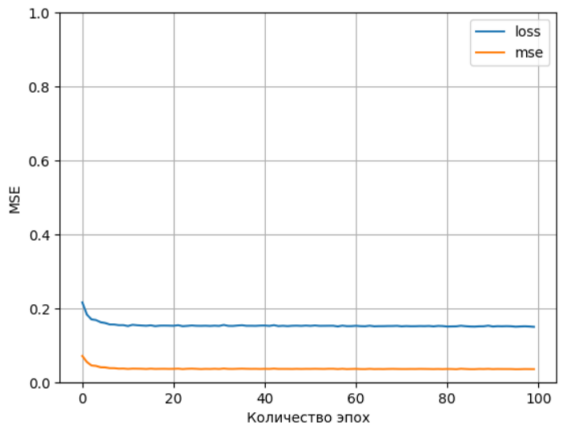
```python
model_ns = Sequential()
#Входной слой с 128 нейронами
model_ns.add(Dense(128, activation = 'relu', input_shape = (X_train_ns.shape[1],)))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(64, activation='relu'))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(32, activation='relu'))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(16, activation='relu'))
model_ns.add(BatchNormalization())
#Выходной слой с одним нейроном
model_ns.add(Dense(1, activation = 'sigmoid'))

#Компиляция
model_ns.compile(optimizer = 'adam', loss = 'mean_absolute_error', metrics = ['mse'])
```

```python
#Оценка ошибки модели на тестовой выборке
print(sqrt(mean_squared_error(y_test_ns, model_ns.predict(X_test_ns))))
```

```
9/9 [==============================] - 0s 2ms/step
0.18645242804582873
```

```python
plot_loss(history)
```



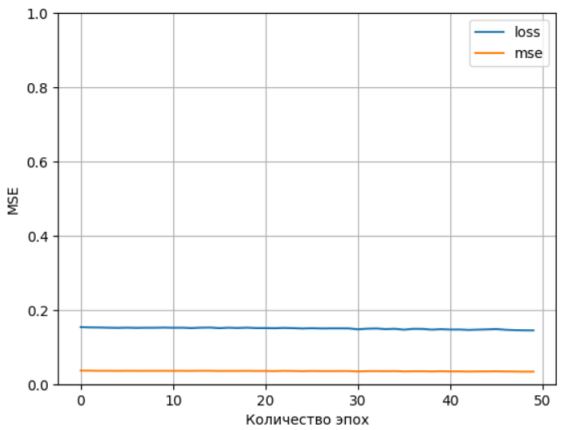

```python
#Архитектура 2 модели
model_ns = Sequential()
model_ns.add(Dense(128, activation = 'relu', input_shape = (X_train_ns.shape[1],)))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(64, activation='relu'))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(32, activation='relu'))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(16, activation='relu'))
model_ns.add(Dense(1, activation = 'sigmoid'))

#Компиляция
model_ns.compile(optimizer = 'adam', loss = 'mean_absolute_error', metrics = ['mse'])
```

```python
#Оценка ошибки 2 модели на тестовой выборке
print(sqrt(mean_squared_error(y_test_ns, model_ns.predict(X_test_ns))))
```

```
9/9 [==============================] - 0s 2ms/step
0.18794809081049207
```

```python
plot_loss(history)
```



ЦЕНТР ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

Нейронная сеть для приложения

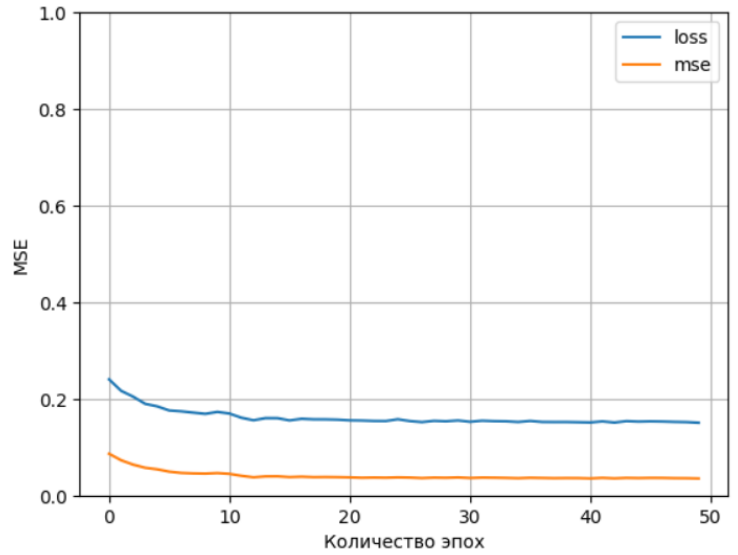
```python
#Архитектура 3 модели
model_ns = Sequential()
model_ns.add(Dense(8, activation = 'relu', input_shape = (X_train_ns.shape[1],)))
model_ns.add(Dropout(0.2))
model_ns.add(Dense(8, activation='relu'))
model_ns.add(Dropout(0.2))
model_ns.add(BatchNormalization())
model_ns.add(Dense(1, activation = 'sigmoid'))

#Компиляция
model_ns.compile(optimizer = 'adam', loss = 'mean_absolute_error', metrics = ['mse'])
```

```python
#Оценка ошибки 3 модели на тестовой выборке
print(sqrt(mean_squared_error(y_test_ns, model_ns.predict(X_test_ns))))
```

```
9/9 [==============================] - 0s 2ms/step
0.1840446228460262
```

```
[ ]  plot_loss(history)
```




```python
import argparse
import keras
#Создаем парсер аргументов командной строки
parser = argparse.ArgumentParser(description='Приложение для прогнозирования соотношения
#Добавляем аргументы для ввода пользователем
parser.add_argument('density', type=float, help='Плотность, кг/м3')
parser.add_argument('elasticity', type=float, help='Модуль упругости, ГПа')
# Загружаем сохраненную модель
model = keras.models.load_model('model_ns.h5')
#Парсим аргументы командной строки
args = parser.parse_args()
density = args.density
```

```python
result = model.predict([[density, elasticity, hardener, epoxy, flash, surface_density,
                         tensile_elasticity, tensile_strength, resin_consumption,
                         weaving_angle, weaving_step, weaving_density]])
#Проверяем, что все аргументы были введены пользователем
if not all(vars(args).values()):
    print('Ошибка: не все аргументы были введены')
else: print('Соотношение матрица-наполнитель:', result[0]) #Выводим результат
```

ЦЕНТР
ДОПОЛНИТЕЛЬНОГО
ОБРАЗОВАНИЯ
МГТУ им. Н.Э. Баумана

do.bmstu.ru