Использование методов машинного обучения для повышения эффективности эксплуатации технологических трубопроводов.

Д.Р. Хуснутдинов damir_khr@mail.ru
Е.О. Смирнова katifreedom1301@gmail.com
В.В.Мокшин vladimir_kgtu@mail.ru

Казанский национальный исследовательский технический университет им. А. Н. Туполева - Каи

Аннотация. *В данной статье основное внимание уделено машинному обучению, и как его можно использовать для повышения эффективности эксплуатации технологических трубопроводов, посредством исследования коррозии метала. Корроозия — самопроизвольное разрушение металлов и сплавов в результате определенного взаимодействия с окружающей средой. Физическое разрушение не является коррозией, а характеризуется понятиями эрозия, простыми словами износ. Причиной появления коррозии является термодинамическая изменчивость конструкционных материалов к воздействию веществ, контактирующих с ними среде. Машинное обучение — это класс методов искусственного интеллекта, характеризующихся не прямым решением задачи, а обучение за счёт применения готовых или изученных ранее решений большого количества похожих задач.*

Ключевые слова: *Анализ, машинное обучения, коррозия металла, линейная регрессия.*

Введение

Исследование проводилось над готовой базой данных с готовыми характеристиками коррозии металла. В базе использовались данные характеристики: Товар, Среда, Расход жидкости, Дозировка, Скорость потока, Частота, Время прохождения жидкости, Давление внутри трубы, Остаточное содержание реагента, Температура, pH, O2 (кислород), Начальная скорость коррозии, Общая скорость коррозии, Питтинг, Текущая эффективность.

1. Для начала была отобрана совокупность основных статистических показателей функционирования объекта исследования.

```
data.head()
```

	Unnamed: 0	Y1	Y2	Y3	X1	X2	X3	X4	X5	X6	...	X11	X12	X13	X14	X
233	234	4	1	103396	14	3	2	2	5.4	3.0	...	11.7	15.2	81	62	35
251	252	16	1	57357	5	2	2	2	5.0	2.5	...	0.0	8.6	57	9	11
4	5	5	3	-410	117	7	3	0	0.0	0.0	...	0.0	0.0	172	0	18
58	59	18	2	1636	14	1	2	2	3.4	3.0	...	2.8	6.9	132	44	12
253	254	16	1	79221	5	2	2	2	3.8	3.0	...	5.6	8.8	77	31	31

5 rows × 24 columns

```
data.tail()
```

	Unnamed: 0	Y1	Y2	Y3	X1	X2	X3	X4	X5	X6	...	X11	X12	X13	X14	X15
92	93	20	1	21330	13	11	2	3	2.7	3.0	...	17.0	21.4	72	27	7.0
393	394	6	1	7226	13	12	2	3	2.0	1.2	...	0.0	6.2	56	11	1.4
158	159	18	2	4092	14	1	2	2	3.4	3.0	...	2.8	5.8	132	43	10.0
84	85	11	1	46512	31	9	2	2	4.0	2.5	...	17.5	16.0	81	55	18.3
20	21	6	2	0	33	0	3	1	0.0	0.0	...	3.8	0.0	96	0	21.0

5 rows × 24 columns

Рис. 1. – Отбор совокупности статистических показателей.

Осуществили просмотр информации по данным: индекс; название колонны; кол-во строк; тип данных.

2. В дальнейшем был произведен анализ выборки и поиск пропусков и аномалий.

```
sns.boxplot(data=X)
```

```
<AxesSubplot:>
```

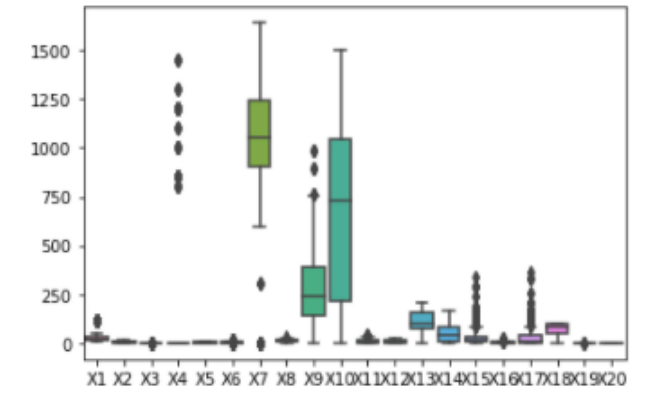


Рис. 2. – Анализ и поиск аномалей.

Точками показаны выбросы. Интервалы отображают минимум и максимум, а прямоугольник медиану. В данных слишком много выбросов.

3. Следующим шагом является вычисление основных статистических характеристик.

Вычисление математического ожидания, среднего квадратичного отклонения и медианы с помощью функции describe. С полученными данными производится поиск эксцессы (асимметрии)

```
f, axes = plt.subplots(2, 10, figsize=(30, 10))
sns.despine(left=True)
for i in range(0, 20):
    sns.distplot(X["X" + str(i+1)], fit=norm, hist=False, ax=axes[i//10,
```

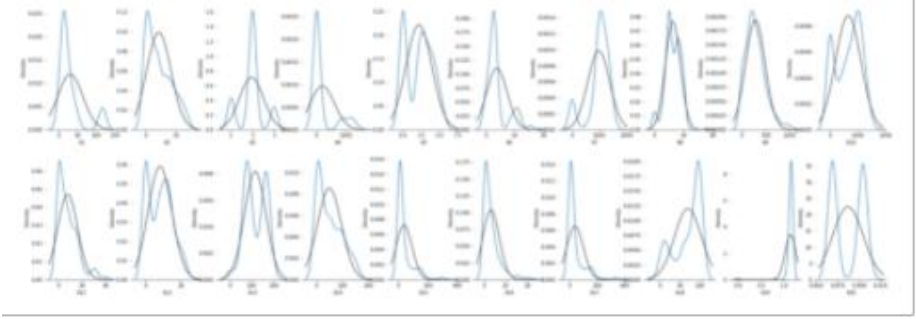


Рис. 3. – Графики эксцессы, ассиметрии. Синим отображён график распределения, чёрным отображено нормальное распределение.

Производится вычисление доверительных интегралов с помощью функции по формуле.

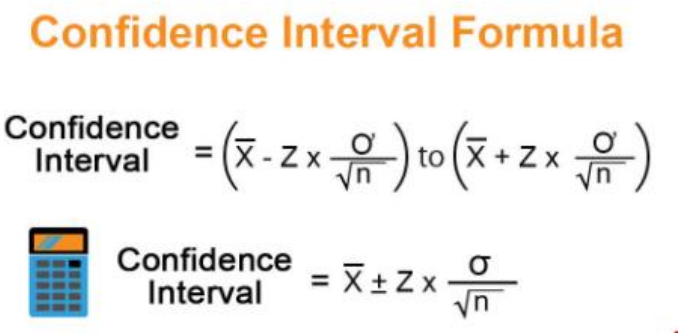


Рис.4 . – Функция для вычисление доверительного интервала по формуле.

```
for i in range(1, 21):
    m, mMinus, mPlus = mean_confidence_interval(X["X" + str(i)], confidence=0.95)
    print("Доверительный интервал для X" + str(i) + " : Среднее: %f с интервалом(%f , %f)" % (m, mMinus, mPlus))
```

```
Доверительный интервал для X1 : Среднее: 30.620000 с интервалом(27.316600 , 33.923400)
Доверительный интервал для X2 : Среднее: 4.280000 с интервалом(3.884829 , 4.675171)
Доверительный интервал для X3 : Среднее: 1.950000 с интервалом(1.894982 , 2.005018)
Доверительный интервал для X4 : Среднее: 190.512500 с интервалом(149.962834 , 231.062166)
Доверительный интервал для X5 : Среднее: 2.205500 с интервалом(2.028346 , 2.382654)
Доверительный интервал для X6 : Среднее: 3.765500 с интервалом(3.410730 , 4.120270)
Доверительный интервал для X7 : Среднее: 972.877750 с интервалом(933.106760 , 1012.648740)
Доверительный интервал для X8 : Среднее: 12.601250 с интервалом(12.091694 , 13.110806)
Доверительный интервал для X9 : Среднее: 282.287500 с интервалом(262.078337 , 302.496663)
Доверительный интервал для X10 : Среднее: 649.537500 с интервалом(604.934482 , 694.140518)
Доверительный интервал для X11 : Среднее: 7.894750 с интервалом(7.062936 , 8.726564)
Доверительный интервал для X12 : Среднее: 8.201500 с интервалом(7.537138 , 8.865862)
Доверительный интервал для X13 : Среднее: 115.465000 с интервалом(110.666828 , 120.263172)
Доверительный интервал для X14 : Среднее: 49.365000 с интервалом(44.768382 , 53.961618)
Доверительный интервал для X15 : Среднее: 37.252500 с интервалом(31.838198 , 42.666802)
Доверительный интервал для X16 : Среднее: 3.151585 с интервалом(2.774162 , 3.529008)
Доверительный интервал для X17 : Среднее: 38.259375 с интервалом(32.233852 , 44.284898)
Доверительный интервал для X18 : Среднее: 69.603750 с интервалом(66.347519 , 72.859981)
Доверительный интервал для X19 : Среднее: 1.132110 с интервалом(1.120594 , 1.143626)
Доверительный интервал для X20 : Среднее: 0.886932 с интервалом(0.885185 , 0.888680)
```

Рис.5 . – Результат: доверительный интеграл.

4. Следует сделать оценку нормального распределения, с помощью их подчинения нормальному закону. Оценка нормальности распределения с помощью функции normaltest.

```
for i in range(1, 20):
stat, p = normaltest(X["X" + str(i)])

print('Оценка нормальности распределения для X' + str(i) + '.
Statistics=%.3f, p=%.3f' % (stat, p))

alpha = 0.05
if p > alpha:
print('Подчиняется нормальному закону распределения')
else:
print('Не подчиняется нормальному закону распределения')
```

Проверяем следуют ли наши признаки нормальному распределению. Данные не подчиняются нормальному закону распределения.

Проверка основана на тестах D'Agostino, R. и Pearson, E. S.

Источник: D'Agostino, R. and Pearson, E. S. (1973), "Tests for departure from normality", Biometrika, 60, 613-622

5. Вычисляем парные коэффеценты линейной корреляции.

```
corrmat = data.corr()
f, ax = plt.subplots(figsize=(18, 14))
sns.heatmap(corrmat, square=True, annot = True);
```

Рис.6 . – Применение функции corr().

Чем белее квадрат, тем больше корреляция. Чем чернее, тем меньше корреляция. Y1 слабо коррелирует с чем-либо.(Наибольшая корреляция с X5 = 0.5). Y2 обратно коррелирует с Y3(-0.56), а также обратно коррелирует с X16(-0.43). Y3 в свою очередь сильно коррелирует с X16(0.95). Некоторые признаки коррелируют друг с другом или являются обратной их корреляцией, их можно было бы исключить.

6. Осуществим факторный анализ.

Факторный анализ (ФА) - это метод исследовательского анализа данных, используемый для поиска влияющих основных факторов или скрытых переменных из набора наблюдаемых переменных. Это помогает в интерпретации данных за счет уменьшения количества переменных. ФА извлекает максимальную общую дисперсию из всех переменных и помещает их в общую оценку.

Основная цель факторного анализа - уменьшить количество наблюдаемых переменных и найти ненаблюдаемые переменные.

Тест сферичности Бартлетта проверяет, коррелируют ли наблюдаемые переменные вообще, используя наблюдаемую корреляционную матрицу с единичной матрицей. Если тест оказался статистически незначимым, не следует использовать факторный анализ.

```
Ввод [41]: from factor_analyzer.factor_analyzer import calculate_bartlett_sphericit
           chi_square_value,p_value=calculate_bartlett_sphericity(data)
           chi_square_value, p_value

Out[41]: (10735.986562684884, 0.0)
```

Рис.7 . – Тест Бартлетта.

В этом тесте Бартлетта p - значение равно 0. Тест является статистически значимым, что указывает на то, что наблюдаемая корреляционная матрица не является тождественной матрицей.

Тест Кайзера-Мейера-Олкина (KMO) измеряет пригодность данных для факторного анализа. Он определяет адекватность для каждой наблюдаемой переменной и для всей модели. KMO оценивает долю дисперсии среди всех наблюдаемых переменных. Идентификатор меньшей доли больше подходит для факторного анализа. Значения KMO находятся в диапазоне от 0 до 1. Значение KMO менее 0,6 считается недостаточным.

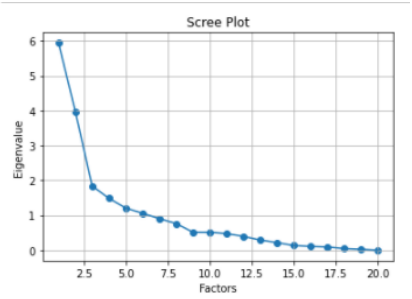
```
Ввод [42]: from factor_analyzer.factor_analyzer import calculate_kmo
           kmo_all,kmo_model=calculate_kmo(X)
           kmo_model

Out[42]: 0.7119544290833609
```

Рис.8 . –Тест Кайзера-Мейера-Олкина

Общий KMO для наших данных больше 0.6, что является отличным показателем. Значит мы можем продолжить факторный анализ.

Для выбора количества факторов можно использовать критерий Кайзера и график осыпи. Оба основаны на собственных значениях.

```
# Create factor analysis object and perform factor analysis

fa = FactorAnalyzer(n_factors = 25, rotation=None)

fa.fit(X)

# Check Eigenvalues

ev, v = fa.get_eigenvalues()

pd.DataFrame(ev)
```



Можно увидеть, что только для шести факторов собственное значение больше единицы. Это означает, что нам нужно выбрать только 6 факторов (или ненаблюдаемых переменных). На рис. 9. – График Осыпи.

7. Нормализация данных. Проведем стандартизацию данны по формуле.

$$z = \frac{x_i - \mu}{\sigma}$$

Рис.10 . – Формула стандартизации данных

```
# применение метода z-score испольщуюя .mean() and .std() methods

def z_score(df):

df_std = df.copy()

for column in df_std.columns:

df_std[column] = (df_std[column] - df_std[column].mean()) /
df_std[column].std()

return df_std
```

```
df_data_standardized = z_score(data)
df_data_standardized
```

	Unnamed: 0	Y1	Y2	Y3	X1	X2	X3	X
296	0.834672	-0.656766	-0.864798	-0.099578	-0.524303	2.169047	0.089331	-0.45939
199	-0.004325	-1.154945	-0.864798	-0.073626	2.540577	-0.069648	-1.697286	2.44709
170	-0.255159	1.502010	-0.864798	0.178854	-0.762352	-0.815880	0.089331	-0.45939
259	0.514642	-0.324647	-0.864798	-0.607311	-0.464791	-0.069648	0.089331	-0.45939
91	-0.938465	-0.656766	-0.864798	-0.384374	-0.524303	1.671559	0.089331	-0.45939
...	...	...	...	...	...	...	...	...
6	-1.673668	-1.154945	-0.864798	0.622596	2.570334	0.925327	-1.697286	1.47745
34	-1.431484	-0.656766	-0.864798	0.581144	-0.494547	-0.069648	0.089331	-0.45697
214	0.125417	-0.490707	0.665818	-0.659713	-0.464791	-1.064624	0.089331	-0.46182
393	1.673668	-0.324647	-0.864798	-0.402825	-0.524303	1.920303	0.089331	-0.45454
210	0.090819	-1.154945	2.196435	-0.777848	-0.464791	-0.815880	-1.697286	1.96227

400 rows × 24 columns

Рис.11 . – Использование метода z-score.

8. Линейная регрессия, используемая в статистике регрессионная модель зависимости одной переменной от другой или нескольких других переменных с линейной функцией зависимости.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import r2_score
Y_train = y_train.iloc[:, 2]
Y_test = y_test.iloc[:, 2]
lr_y3 = LogisticRegression(multi_class="auto")
lr_y3.fit(x_train, Y_train)
print("Уравнение линейной регресии:")
y_string = "y = 0"
for i in range(np.shape(lr_y3.coef_[0])[0]):
    y_string += str(round(lr_y3.coef_[0][i], 4))+"*x"+str(i)+" + "
y_string += str(round(lr_y3.intercept_[0], 4));
print(y_string)
```

```
Уравнение линейной регресии:
y = 0-0.0219*x0 + -0.4875*x1 + -0.1751*x2 + 0.5002*x3 + -0.186*x4 + -0.
0837*x5 + 0.2582*x6 + 0.5132*x7 + 0.0518*x8 + -0.1658*x9 + 0.2583*x10 +
-0.6493*x11 + 0.2235*x12 + 0.2863*x13 + 0.2658*x14 + 0.6131*x15 + 0.223
9*x16 + 0.0407*x17 + 0.0121*x18 + -0.2076*x19 + -0.3892
```

Рис.12 . – Тренируем модель линейной регрессии по всем признакам.

```
print("Точность линейной регрессии в R^2 формате для тренировочного сета
```

```
Точность линейной регрессии в R^2 формате для тренировочного сета: 0.96
44151277660765
```

```
print("Точность линейной регрессии в R^2 формате для тестового сета: {}"
```

```
Точность линейной регрессии в R^2 формате для тестового сета: 0.8592231
49477384
```

Рис.13 . – Тренировочный и тестовый сеты.

Точность нашей модели равна 93%. Очень неплохо. Что если попробовать при тренировке использовать только признаки, выбранные методами отбора признаков.

```
lr2 = LogisticRegression(multi_class="auto")
lr2.fit(x_train[:, [2, 3, 5, 6, 8, 13, 15, 17]], y_train.iloc[:, 2])
print("Точность линейной регрессии в R^2 формате: {}".format(r2_score(Y_
```

```
Точность линейной регрессии в R^2 формате: 0.636762733044856
```

Рис.14 . – Точность Линейной регрессии отобранных признаков.

Гораздо хуже(79%). Выборка факторов бесполезна в данном случае, ведь каждый фактор важен для тренировки. Перейдём к графическому представлению.

Теперь мы можем использовать нашу модель для вычисления y3, y2 и y1 для тестовых тестов, чтобы их сравнить.

Два верхних графика: Синим отображено то, что выдаёт модель, а красным - проверочные значения. Чем ближе синие значения к красным, тем лучше.

Два нижних графика отображают разницу между синими и красными точками на верхних графиках (Результаты представлены на рисунках 15,16 и 17).

```
(-21.0, 21.0)
```

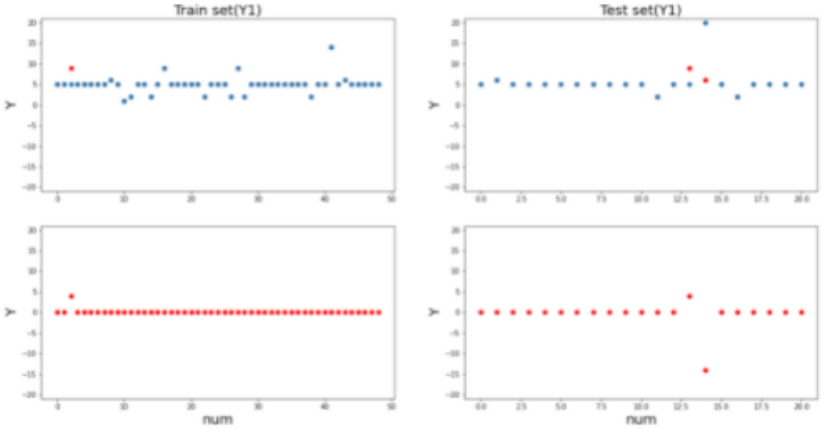


Рис.15 . – Прогнозирование линейной регрессии для y1.

```
(-4.0, 4.0)
```

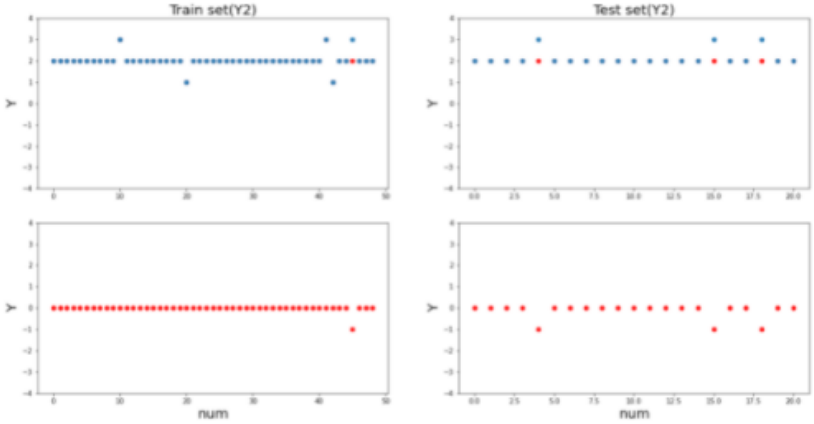


Рис.16 . – Прогнозирование линейной регрессии для y2.

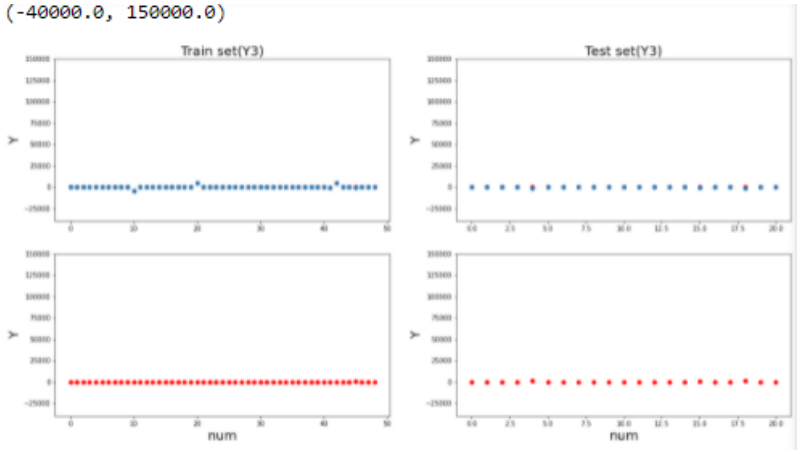


Рис.17 . – Прогнозирование линейной регрессии для y3.

Итоги линейной регрессии для y1, y2 и y3

```
pd.DataFrame({'Индексы колонн': [sfs_y1_factors, sbs_y1_factors, np.wher
              'Тренировочная точность': [sfs_y1_score_train, sbs_y1_scor
              'Тестовая точность': [sfs_y1_score_test, sbs_y1_score_test
                  index=['SFS', 'SBS', 'Генетический алгоритм', 'Корреля
```

	Индексы колонн	Тренировочная точность	Тестовая точность
SFS	(0, 1, 2, 3, 4, 6, 7, 8, 16, 17)	0.578571	0.416667
SBS	(0, 1, 2, 3, 4, 7, 8, 9, 14, 17)	0.610714	0.466667
Генетический алгоритм	[1, 2, 4, 5, 6, 8, 11, 12, 13, 14, 15, 16, 17,...	0.675000	0.675000
Корреляционный анализ	[0, 2, 3, 4, 5, 8, 9, 10, 11, 12, 13, 14, 16, ...	0.700000	0.508333

Рис.18 . –Таблица точностей для y1.

```
pd.DataFrame({'Индексы колонн': [sfs_y2_factors, sbs_y2_factors, np.wher
              'Тренировочная точность': [sfs_y2_score_train, sbs_y2_scor
              'Тестовая точность': [sfs_y2_score_test, sbs_y2_score_test
                 index=['SFS', 'SBS', 'Генетический алгоритм', 'Корреля
```

	Индексы колонн	Тренировочная точность	Тестовая точность
SFS	(0, 1, 2, 3, 4, 6, 7, 8, 16, 17)	0.696429	0.700000
SBS	(0, 1, 2, 3, 4, 7, 8, 9, 14, 17)	0.739286	0.741667
Генетический алгоритм	[1, 2, 3, 5, 6, 8, 11, 12, 14, 15, 16, 17, 19]	0.821429	0.783333
Корреляционный анализ	[0, 1, 2, 4, 6, 7, 8, 9, 10, 11, 12, 15, 17, 1...	0.789286	0.766667

Рис.19 . –Таблица точностей для y2.

```
pd.DataFrame({'Индексы колонн': [sfs_y3_factors, sbs_y3_factors, np.wher
              'Тренировочная точность': [sfs_y3_score_train, sbs_y3_scor
              'Тестовая точность': [sfs_y3_score_test, sbs_y3_score_test
                 index=['SFS', 'SBS', 'Генетический алгоритм', 'Корреля
```

	Индексы колонн	Тренировочная точность	Тестовая точность
SFS	(1, 2, 3, 5, 6, 8, 13, 15, 17, 19)	0.864380	0.639182
SBS	(2, 3, 5, 6, 8, 13, 14, 15, 16, 17)	0.818459	0.638785
Генетический алгоритм	[0, 4, 5, 6, 8, 9, 10, 11, 12, 13, 15, 16, 17,...	0.924103	0.789644
Корреляционный анализ	[2, 4, 5, 6, 9, 11, 12, 14, 15, 17, 19]	0.881055	0.713498

Рис.20 . –Таблица точностей для y3.

Вывод

С помощью готовых данных в безе данных и машинного обучения в статье мы смогли обработать массив данных, на освнове анализа мы получили необходимые значения.

Использование машинного обучения на производстве принесет большую пользу для производителей, ведь они смогут не только быстрее находить места возможной коррозии, но и определиться с каким материалом будет выгоднее всего работать.

Проведение данной работы возможно не только с данными связанными с коррозией металла. Можно разработать модель помогающую прогнозировать разрушение друзих природных материалов под воздействием совершенно других веществ и природных явлений с течением времени. Это может помочь оптимизоровать использование привычных материалов, также открыть новые спомобы их использования и воздействя с разными строительными ресурсами.

Литература

1. Якупов Д.Т., Мокшин В.В., Кирпичников А.П. Сравнение методов спектральной кластеризации графовых моделей трубопроводных систем. URL: https://elibrary.ru/item.asp?id=41240274 (дата обращения 19.12.2021)
2. Мокшин В.В., Спиридонова А.В., Спиридонов Г.В. Применение математических моделей и алгоритмов при прогнозировании потребления водных ресурсов. URL: https://elibrary.ru/item.asp?id=46452348 (дата обращения: 18.12.2021).
3. Салихова Э.И., Мокшин В.В., Кирпичников А.П., Тутубалин П.И., Михайлова. Мониторинг буровых работ с использованием средств распознавания образов. URL: https://elibrary.ru/item.asp?id=32683912 (дата обращения: 20.12.2021).
4. Зафиевский А.В., Короткин А.А,, Лататуев А.Н Базы данных. Учебное пособие. Ярославский государственный университет, 2012 Прикладная информатика.
5. Карпова И.П. Базы данных. Курс лекций. Москва 2013 Информатика и вычислительная техника.