ИНФОРМАТИКА

УДК 658.8:004.6 DOI 10.25730/VSU.0536.24.007

Сравнение двух методов анализа данных по продажам в сети магазинов Rossmann

Шатров Анатолий Викторович[1], Левин Михаил Наумович[2]

[1]доктор физико-математических наук, главный научный сотрудник кафедры ЭВМ,
Вятский государственный университет. Россия, г. Киров; профессор физико-механического института,
Санкт-Петербургский политехнический университет.
Россия, г. Санкт-Петербург. ORCID: 0000-0002-5295-571X. E-mail: shatrov@vyatsu.ru
[2]кандидат физико-математических наук, доцент кафедры прикладной математики и информатики,
Вятский государственный университет. Россия, г. Киров. E-mail: usr00227@vyatsu.ru

Аннотация. В данной работе производится сравнение двух методов прогнозирования продаж по данным статистики европейской торговой сети Rossmann. В качестве методов сравнения выбраны метод множественной линейной регрессии (MultipleLinearRegression – MLR) и модифицированный метод случайного леса (RandomForest). В качестве инструментальной базы расчетов используется среда разработки Python. Предварительная обработка базы данных по типам магазинов, временным интервалам работы сети, состоянию спроса потребителей в зависимости от различных факторов позволила сформировать целевую функцию объема продаж Sales. Обсуждается модифицированная постановка задачи метода случайного леса в контексте формирования ансамбля деревьев решений. Это достигается путем обучения алгоритма Random Forest по обучающей выборке и последующего осреднения по ансамблю. По результатам расчета сделан вывод о преимуществе модели случайного леса на основании значений критериев R^2 (коэффициент детерминации) и RMSE (средняя квадратичная ошибка).

Ключевые слова: множественная линейная регрессия, метод случайного леса, прогнозирование, Python.

Введение. Ранее нами был осуществлен эконометрический анализ данных торговой статистики сети магазинов Rossmann[1], представленной на платформе Kaggle (https://www.kaggle.com/), позволила выявить основные закономерности в зависимости целевой функции Sales от факторов, представленных в исходной базе данных. Методы предварительной обработки данных, дисперсионного и корреляционного анализа [1–3] дают возможность выделить наиболее значимые факторы, влияющие на целевую функцию Sales, представляющую объемы продаж торговой сети. В качестве моделей, обеспечивающих оптимизацию целевой функции, выбраны модель множественной линейной регрессии (MultipleLinearRegression) и модель случайного леса (RandomForest) [4]. Алгоритмы расчета для указанных моделей реализованы в среде разработки Python [10]. Критериями адекватности при сравнении этих моделей выбраны коэффициент детерминации R^2 и средняя квадратичная ошибка RMSE.

1. MLR – множественная линейная регрессия. Рассматриваемая модель MLR применяется в анализе базы данных используемой статистики продаж для классификации факторов и последующего прогноза целевой функции Sales.

Параметры регрессионной модели рассчитываются с помощью метода наименьших квадратов (OrdinaryLeastSquare – OLS) [3; 12]. Как известно, метод OLS представляет собой основную процедуру в контексте машинного обучения с учителем для прогноза значений целевой переменной, зависящей от заданного числа предикторов. Для реализации МЛР программными средствами Python используется приложение sklearn [10]. Для этого необходимо вызвать функцию LinearRegression с помощью команды:

classsklearn.linear_model.LinearRegression(fit_intercept = True , normalize = False , copy_X = True , n_jobs = 1)

В данной команде используется алгоритм обучения модели fit (X_train, Y_train) [7; 8], где X_train представляет набор факторов-предикторов (независимых переменных), а Y_train – отклик

[1] Статья авторов «Эконометрический анализ торговой статистики сети магазинов Rossmann» в предыдущем выпуске журнала.

(целевая функция) Sales. Для получения прогноза в работе используются два метода: MLR (LinearRegression) и метод случайного леса RF (Random Forest [4; 9]). Рассмотрим реализацию MLR, в терминах параметров вызываемой функции LinearRegression. В процессе реализации вызываемая функция вычисляется для двух выборок: X_train, Y_train, представляющих значения предикторов и отклика соответственно. Для непосредственного вычисления используется команда fit (X_train, Y_train).

Перечень основных параметров вызываемой функции LinearRegression (…) перечислен в таблице 1.

Таблица 1

Параметры функции LinearRegression ()

Параметр	Тип	Описание
fit_intercept	bool	Признак центрированности данных: по умолчанию True
normalize	bool	Этот параметр игнорируется, если для fit_intercept = False. Если True, независимые переменные X будут нормализованы до регрессии путем вычитания среднего значения
copy_X	bool	Если True, X будет скопирован; иначе он может быть перезаписан
n_jobs	Int	Количество заданий для вычисления

Программа расчета в среде Python представлена в Приложении А.

Для проверки адекватности и точности модели вычисляются критерии тестирующей выборки – коэффициент детерминации R^2 и средняя квадратичная ошибка RMSE. Вычисленные значения R^2=0.5432, RMSE = 1281.83.

Значения критериев R^2 и RMSE свидетельствуют, что модель MLR не является адекватной и точной, поэтому не может быть принята для предсказания значений целевой функции Sales.

На рисунке 1 приводится диаграмма рассеивания значений целевой функции.

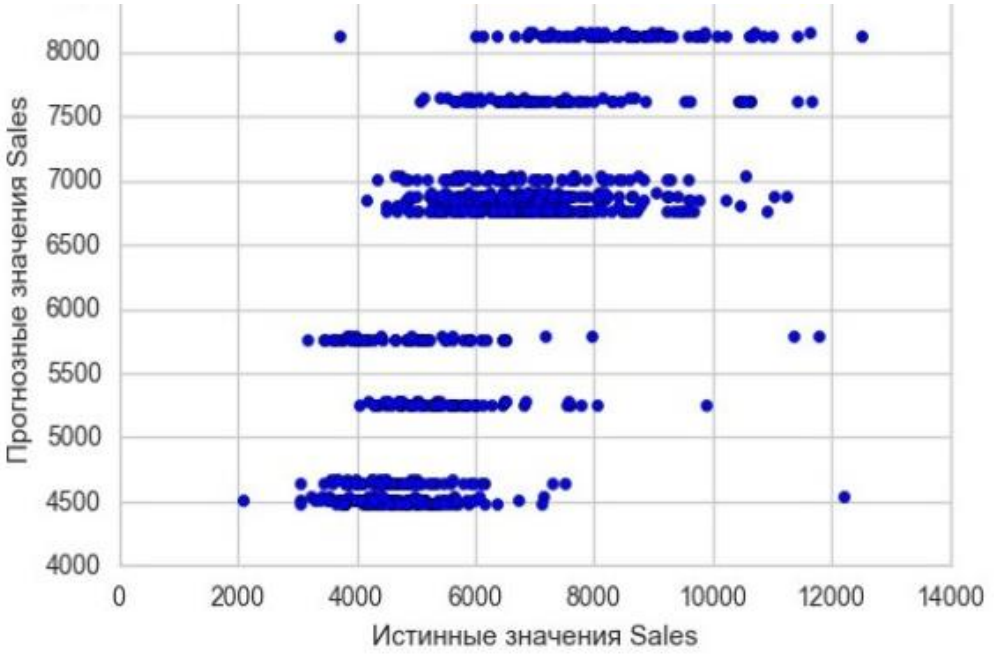

Рис. 1. Диаграмма рассеивания для прогноза линейной регрессии

При этом необходимо понимать, что существуют методы обобщения линейной регрессии, в частности, модели с регуляризацией или модификацией квадратичного функционала вOLS. Регуляризованной регрессией (или регрессией со штрафом) является модель, в которой на регрессионные коэффициенты накладывается некоторый штраф с целью улучшить ее способности к обобщению.

В случае обычной регрессии целью является достижение наилучшей аппроксимации заданной линейной зависимости (по обучающей выборке), что приводит к переобучению (overfitting). В этом случае метод хорошо работает на обучающей выборке и плохо работает на примерах, не участвовавших в обучении.

Математически регуляризацию можно описать следующим образом. Имеем набор данных $(x_1, y_1), \ldots, (x_n, y_n)$, где $x_i \in R^n$ и $y_i \in \{-1, 1\}$. Целевая функция имеет вид $f(x) = w^T x + b$ с параметрами $w \in R^m$

и свободным членом $b \in$ R. Для того, чтобы сделать прогноз, мы смотрим на знак функции *f(x)*. Найдем параметры модели, минимизируя тренировочную ошибку:

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f(x_i)) + \alpha R(w) \quad (1)$$

где *L* – функция потерь, *R* – функция регуляризации, $\alpha > 0$.

Наиболее популярные виды функции *R*:

– $R(w) := \frac{1}{2}\sum_{i=1}^{n} w_i^2$ (L1)

– $R(w) := \sum_{i=1}^{n} |w_i|$ (L2)

– $R(w) := \frac{p}{2}\sum_{i=1}^{n} w_i^2 + (1 - p)\sum_{i=1}^{n} |w_i|$ (Elastic Net) – выпуклая комбинация L1 и L2 (рис. 2).

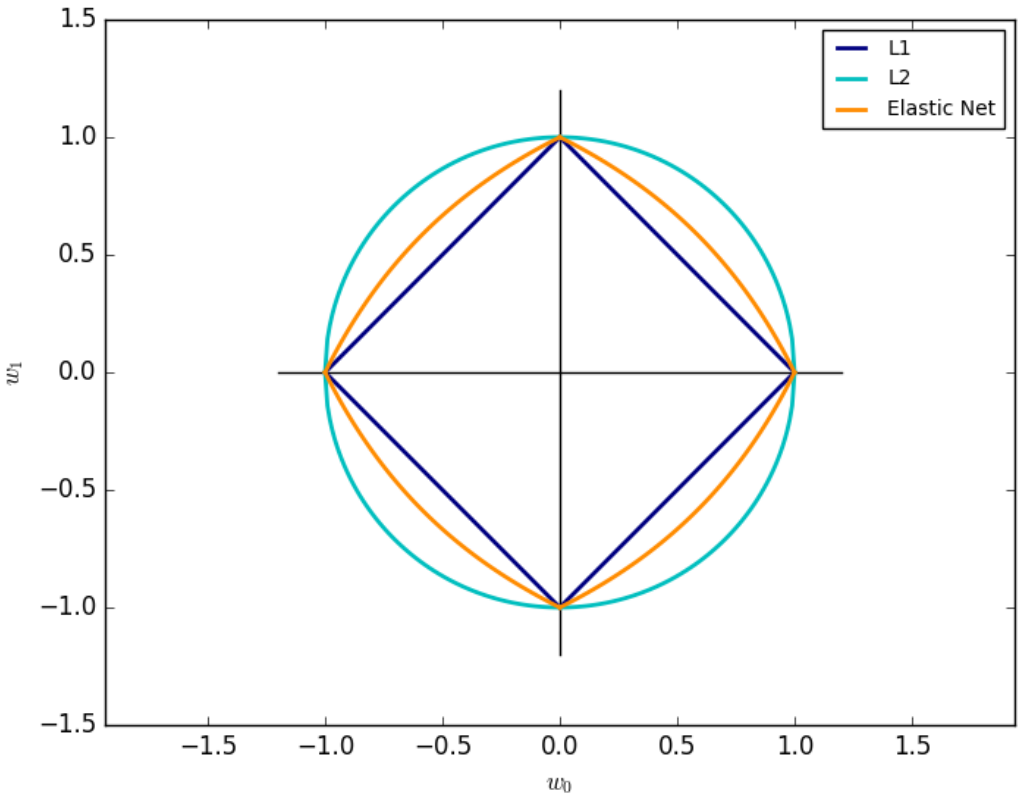

Рис. 2. Функция регуляризации *R(w)*

Чаще всего используются два способа регуляризации (штрафа), применяемых в случае регрессии: L1 и L2. Для L1 функция потерь примет вид:

$$\sum_i^n (y_i - f_i(x))^2 + \alpha \sum_i^n |w_i| -> min_x \quad (2)$$

В таком случае одновременно с уменьшением ошибки мы уменьшаем и значения коэффициентов (по их модулю). Для штрафа L2 получим:

$$\sum_i^n (y_i - f_i(x))^2 + \alpha \sum_i^n w_i^2 -> min_x \quad (3)$$

α в обоих случаях – параметр регуляризации, контролирующий величину штрафа. Чем меньше его значение, тем меньше штраф. Модели, имеющие в своем составе штрафы L1 и L2, называются Lasso[2] регрессией и Ridge регрессией (или гребневой) соответственно. Оба алгоритма будут давать меньшие значения весовых коэффициентов, нежели обычный метод наименьших квадратов.

Однако использование обобщения МЛР за счет регуляризации не избавляет от проблемы переобучения. Одним из методов для решения этой проблемы является использование третьего набора данных – валидационного. Как бы то ни было, разбивая исходный набор данных на три части, мы сильно снижаем количество образцов, которые можем использовать для обучения модели, потенциально снижая ее качество.

2. RF – Модель RandomForest. Особенность анализа многомерных статистических выборок для получения прогноза целевой функции методом построения регрессионной зависимости заключается не только в подборе коэффициентов (параметров) регрессии, но и в обоснованном выборе предикторов этой зависимости. Такая задача решается в процессе классификации исходной выборки независимых факторов. Конечного результата можно достичь, используя метод случайного леса

[2] Least absolute shrinkage and selection operator.

(RF – RandomForest) [4; 9], сочетающего возможность решения задачи регрессии для отклика (целевой функции) и задачи классификации предикторов (определяющих факторов). Метод RF представляет обобщение алгоритма решающих деревьев (DecisionTree), который использует ансамбль деревьев для улучшения качества классификации или регрессии. Суть обобщающего метода RF состоит в том, что производится случайный перебор деревьев, составленных из обучающей выборки. Затем производится ансамблирование (комбинация) случайных подмножеств. Полученный ансамбль позволяет уменьшить эффект переобучения и повысить качество предсказаний. Реализация алгоритма выполняется в следующей последовательности:

1. Формируем случайную выборку из факторов (объектов) исходного набора данных.

2. Составляем случайную выборку из исходного набора признаков обучающей выборки.

3. На основании выполненных этапов 1, 2 строим дерево решений для оценки наилучшего признака при разбиении данных на каждом уровне дерева.

4. Реализация алгоритма или построение ансамбля заключается в повторении шагов 1–3 для каждого дерева.

Уравнение регрессии для предсказания целевой функции представляется в результате усреднения по ансамблю решений $T_i(x)$. Прогноз новых значений представляется следующим образом:

$$y^X = \frac{\sum_{i=1}^{X} T_i(x)}{X} \quad (4)$$

Выполненное усреднение (4) уменьшает дисперсию отдельных деревьев решений.

Модель RandomForest реализуется в Python с помощью функции sklearn.ensemble.Random ForestRegressor [5; 6; 11]:

class sklearn.ensemble.RandomForestRegressor (n_estimators=Int, criterion='mse', max_depth= None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score= False, n_jobs=1, random_state=None, verbose=0, warm_start=False).

Алгоритм производит на этапе обучения классификацию объектов (факторов) модели. Как отмечалось выше, это является главным преимуществом ансамблевого метода RF, что в конечном итоге позволяет оценить важность каждого фактора для классификации объектов. Это можно сделать, используя атрибут feature_importances модели, который возвращает массив значений, отражающих важность каждого признака. Более важные признаки будут иметь более высокие значения.

Ниже в таблице 2 приводится перечень и описание основных параметров функции sklearn.ensemble.RandomForestRegressor. Принятая в качестве стандарта среды разработки Python реализация алгоритма модели RandomForest предполагает такое использование параметров, при котором можно их варьировать, осуществляя настройку при выполнении алгоритма. Для этого достаточно менять, например, количество деревьев в ансамбле, количество объектов и признаков в каждом подмножестве, критерий информативности, минимальное количество объектов, при котором осуществляется разбиение, количество конечных узлов, максимальную глубину деревьев и т. д.

Вычисленные значения коэффициентов влияния приведены в таблице 3. Структура таблицы 3 демонстрирует очередность факторов по степени важности (коэффициентов влияния) в порядке их убывания.

Таким образом можно установить основные гиперпараметры модели – число деревьев и максимальную глубину дерева. В таблице 4 на основании вычислительного эксперимента приводятся оптимальные значения этих ключевых параметров алгоритма Random Forest. В качестве критериев при осуществлении подбора параметров использовались коэффициент детерминации R^2 и средняя квадратичная ошибка RMSE.

Программа выполнения алгоритма модели RandomForest в среде разработки Python приводится в Приложении В.

Таблица 2

Параметры функции RandomForestRegressor ()

Параметр	Тип	Описание
n_estimators	Int	Число деревьев – при увеличении количества деревьев улучшается качество модели, время построения модели увеличивается.
criterion	string	Функция для измерения оценки качества разбиения деревьев.
max_features	Int, float, string or None	Количество признаков выбора расщепления – для задачи регрессии равен n/3. Является очень важным параметром. При увеличении данного параметра время построения модели увеличивается.

Окончание табл. 2

Параметр	Тип	Описание
min_samples_split	Int, float	Минимальное количество объектов, при котором осуществляется разбиение. При увеличении качество снижается, а время построения модели сокращается.
min_samples_leaf	Int, float	Минимальное количество объектов в листьях, для задач регрессии рекомендуется использовать значение 5.
max_depth	Intor None	Максимальная глубина дерева.
min_weight_fraction_leaf	float	Минимальная взвешенная доля суммы весов (всех входных выборок), необходимых для конечного узла.
max_leaf_nodes	IntorNone	Количество конечных узлов. Если None, то неограниченное количество конечных узлов.
bootstrap=True	Bool (True or False)	Параметр построения решающего дерева. Если False, то для построения каждого дерева используется весь набор данных.
oob_score=False	Bool (True or False)	Параметр дополнительной оценки обобщенного признака.
n_jobs=1	Int	Количество заданий, выполняемых параллельно.
warm_start=False	Bool (True or False)	Параметр настройки. Если True, необходимо повторно использовать решение предыдущего вызова для подгонки алгоритма.

Для подбора этих параметров использовался метод ручной настройки – подбор осуществлялся экспериментально.

Таблица 3

Значение коэффициентов влияния факторов на эффективность модели прогнозирования

Фактор	Значение коэффициента
CompetitionDistance	0.194650
Store	0.174714
Promo	0.135008
DayOfWeek	0.071729
CompetitionOpenSinceYear	0.058781
CompetitionOpenSinceMonth	0.058551
CompetitionOpen	0.049446
Day	0.043992
WeekOfYear	0.037136
StoreType	0.036059
Promo2SinceYear	0.029928
Assortiment	0.029330
Promo2SinceWeek	0.024404
PromoInterval	0.014829
PromoOpen	0.014714
Month	0.011000
SchoolHoliday	0.005532
Year	0.004418
Promo2	0.002683
StateHoliday	0.001611
IsPromoMonth	0.001484

Таблица 4

Параметры модели RF

n_estimators	3000
maxdepth	10

В результате использования алгоритма Random Forest для прогноза основные критерии адекватности принимают значения R^2 = 0.9100 и RMSE = 0.16582. Таким образом, можно утверждать, что модель Random Forest является адекватной и предсказывает результат прогнозирования существенно лучше, чем линейная регрессия.

Заключение. Для прогнозирования динамики изменения целевой функции Sales были использованы две модели: модель множественной регрессии и модель случайного леса RF. В резуль-

тате сравнения для выбранных данных была выбрана модель прогнозирования, которая обладает низкой ошибкой и коэффициентом детерминации, близким к 1. Выбранная модель использует модифицированный алгоритм RF, который относится к методам обучения с учителем и является ансамблевым методом. Модель случайного леса является актуальным и адекватным алгоритмом машинного обучения, способным выполнять функции классификации факторов-предикторов и прогнозирования целевых функций. Вместе с тем алгоритм RF имеет как свои преимущества, так и недостатки. К несомненным преимуществам этого алгоритма относятся следующие свойства:

1. Модель RF может использоваться для задач классификации и регрессии в части прогнозирования целевых функций.

2. Модель RF позволяет обрабатывать данные с большим количеством признаков.

3. Модель RF является достаточно устойчивой к переобучению.

4. Модель дает возможность измерять значимость каждого признака для задачи.

При этом модели RF присущи следующие недостатки:

1. Модель может быть сложной для интерпретации, так как она оперирует большим числом операторов и параметров, принадлежащих множеству деревьев решений.

2. Имеется необходимость в настройке параметров модели, приведенных в таблице 2.

3. Для обучения модели RF может потребоваться существенно больше времени по сравнению с MLR, особенно при использовании большого количества деревьев.

Список литературы

1. *Магнус Я. Р., Катышев П. К., Пересецкий А. А.* Эконометрика. Начальный курс. М. : Дело, 2007. 504 с.

2. *Нестеров С. А.* Базы данных. Интеллектуальный анализ данных : учебное пособие. СПб. : Изд-во Политехн. ун-та, 2011. 272 с.

3. *Box G., Jenkins G.* Time series analysis: forecasting and control. John Wiley and Sons, 2008. P. 748.

4. *Breiman Leo*. Random Forests // Machine Learning. 2001. Т. 45. № 1. Pp. 5–32.

5. *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning, 2nd ed. Springer, 2009. 763 p.

6. *Fayyad M., Piatetsky-Shapiro G., Smyth P.* From Data Mining to Knowledge Discovery in Databases. URL: https://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf (дата обращения: 01.02.2023).

7. NumPyUserGuide (Release 1.14.0). URL: http://docs.scipy.org/doc/numpy/user/ (дата обращения: 11.05.2023).

8. Pandas: powerful Python data analysis toolkit (Release 0.23.0). URL: http://pandas.pydata.org/pandas-docs/stable/ (дата обращения: 11.05.2023).

9. *Piatetsky-Shapiro G*. CRISP-DM, still the top methodology for analytics, data mining, or data science projects. 2014. URL: http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-datascience-projects.html (дата обращения: 25.05.2023).

10. Scikit-learnuserguide (Release 0.19.1). URL: http://scikit-learn.org/stable/user_guide.html (дата обращения: 11.05.2023).

11. Seaborn: Annotated heatmaps (Release 0.9.0). URL: http://seaborn.pydata. org/examples/heatmap_annotation.html (дата обращения: 11.05.2023).

12. Statsmodels: statistics in Python (Release 0.9.0). URL: http://www.statsmodels.org/stable/index.html (дата обращения: 11.05.2023).

Приложение А

Программный код для построения модели линейной регрессии

```
rossmann_dic = dict(list(train.groupby('Store')))
test_dic = dict(list(test_df.groupby('Store')))
submission = Series()
scores = []
for i in test_dic:
        # current store
        store = rossmann_dic[i]
                # define training and testing sets
X_train = store.drop(["Sales","Store"],axis=1)
Y_train = store["Sales"]
X_test =test_dic[i].copy()
store_ids = X_test["Id"]
X_test.drop(["Id","Store"], axis=1,inplace=True)
        # Linear Regression
lreg = LinearRegression()
lreg.fit(X_train, Y_train)
        Y_pred2=lreg.predict(X_train)
scores.append(lreg.score(X_train, Y_train))
Y_train=list(Y_train)
plt.scatter(Y_train,Y_pred2)
plt.xlabel(u'Истинные значения Sales')
plt.ylabel(u'Прогнозные значения Sales')
plt.title(u'Диаграмма рассеяния для прогноза линейной регрессии')
error=0
for i in range(len(Y_train)):
        error=(abs(Y_pred2[i]-Y_train[i])/Y_train[i])
train_error_bay=error/len(Y_train)*100
print("Train error = " +'{}'.format(train_error_bay)+ " percent in SVM")
        error = mean_squared_error(Y_train, Y_pred2)
print("MSE="+'{}'.format(error))
RR=r2_score(Y_train, Y_pred2)
print("RR="+'{}'.format(RR))
RMSE=0
R=0
for i in range(len(Y_train)):
        R=(Y_pred2[i]-Y_train[i])**2
RMSE=math.sqrt(R/len(Y_train))
print("RMSE="+'{}'.format(RMSE))
```

Приложение В

Программный код для построения модели Randomforest

```
test = pd.read_csv(io.StringIO(uploaded['test.csv'].decode('utf-8')))
dfdf =newnew..copycopy()()
test.fillna(1, inplace=True)
df = df[df["Open"] != 0]
df = df[df["Sales"] > 0]
df['log_sales'] = np.log(df['Sales'])
df = pd.merge(df, store, on='Store')
test = pd.merge(test, store, on='Store')
df.fillna(0,inplace=True)
test.fillna(0,inplace=True)
dfdf[["StateHoliday""StateH ] = df["StateHoliday"].map({0: 0, "0": 0, "a": 1, "b": 1, "c": 1})
test["StateHoliday"] = test["StateHoliday"].map({0: 0, "0": 0, "a": 1, "b": 1, "c": 1})
df['StateHoliday'] = df['StateHoliday'].astype(float)
test['StateHoliday'] = test['StateHoliday'].astype(float)
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.20, random_state=7)
```

```
X_test=test.drop(['Id','Store','Date','CompetitionOpenSinceYear','Promo2SinceYear','PromoInterval'] , axis = 1)
rf = RandomForestRegressor(n_estimators=3000,max_depth=10)
rf.fit(X_train,y_train)
rf_pred = rf.predict(X_val)
rmse_rf = np.sqrt(mean_squared_error(y_val,rf_pred))
rmse_rf
fig, ax = plt.subplots(1, 1, figsize=(15,8))
sns.barplot(x=0, y=1, color="salmon", data=feature_importance, edgecolor="black")
plt.xlabel("Значения факторов ")
plt.ylabel("Факторы")
```

Comparison of two methods of analyzing sales data in the Rossmann store chain

Shatrov Anatoly Viktorovich[1], Levin Mikhail Naumovich[2]

[1]Doctor of Physical and Mathematical Sciences, Chief Researcher of the Computer Department, Vyatka State University. Russia, Kirov; professor of the Institute of Physics and Mechanics, St. Petersburg Polytechnic University. Russia, St. Petersburg. ORCID: 000-0002-5295-571X. E-mail: shatrov@vyatsu.ru

[2]PhD in Physical and Mathematical Sciences, associate professor of the Department of Applied Mathematics and Computer Science, Vyatka State University. Russia, Kirov. E-mail: usr00227@vyatsu.ru

Abstract. This paper compares two sales forecasting methods based on statistics from the European Rossmann retail chain. The method of multiple linear regression (MultipleLinearRegression – MLR) and the modified random forest method (randomForest) were chosen as comparison methods. The Python development environment is used as a calculation tool. Preliminary processing of the database by types of stores, time intervals of the network, and the state of consumer demand, depending on various factors, allowed us to form the target function of Sales volume. A modified formulation of the problem of the random forest method is discussed in the context of the formation of an ensemble of decision trees. This is achieved by training the Random Forest algorithm on a training sample and then averaging over the ensemble. Based on the calculation results, it is concluded that the random forest model is advantageous based on the values of the criteria R2 (coefficient of determination) and RMSE (mean square error).

Keywords: multiple linear regression, random forest method, forecasting, Python.

References

1. *Magnus Ya. R., Katyshev P. K., Pereseckij A. A. Ekonometrika. Nachal'nyj kurs* [Econometrics. The initial course]. M. Delo (Business), 2007. 504 p.

2. *Nesterov S. A. Bazy dannyh. Intellektual'nyj analiz dannyh : uchebnoe posobie* [Databases. Data mining : textbook]. SPb. Polytechnic University Publishing House, 2011. 272 p.

3. *Box G., Jenkins G.* Time series analysis: forecasting and control. John Wiley and Sons, 2008. P. 748.

4. *Breiman Leo*. Random Forests // Machine Learning. 2001. Vol. 45. No. 1. Pp. 5–32.

5. *Hastie T., Tibshirani R., Friedman J.* The Elements of Statistical Learning, 2nd ed. Springer, 2009. 763 p.

6. *Fayyad M., Piatetsky-Shapiro G., Smyth P.* From Data Mining to Knowledge Discovery in Databases. Available at: https://www.kdnuggets.com/gpspubs/aimag-kdd-overview-1996-Fayyad.pdf (date accessed: 01.02.2023).

7. NumPyUserGuide (Release 1.14.0). Available at: http://docs.scipy.org/doc/numpy/user/ (date accessed: 11.05.2023).

8. Pandas: powerful Python data analysis toolkit (Release 0.23.0). Available at: http://pandas.pydata.org/pandas-docs/stable/ (date accessed: 11.05.2023).

9. *Piatetsky-Shapiro G*. CRISP-DM, still the top methodology for analytics, data mining, or data science projects. 2014. Available at: http://www.kdnuggets.com/2014/10/crisp-dm-top-methodology-analytics-data-mining-datascience-projects.html (date accessed: 25.05.2023).

10. Scikit-learnuserguide (Release 0.19.1). Available at: http://scikit-learn.org/stable/user_guide.html (date accessed: 11.05.2023).

11. Seaborn: Annotated heatmaps (Release 0.9.0). Available at: http://seaborn.pydata.org/examples/heatmap_annotation.html (date accessed: 11.05.2023).

12. Statsmodels: statistics in Python (Release 0.9.0). Available at: http://www.statsmodels.org/stable/index.html (date accessed: 11.05.2023).