1. Знакомство и участие в разработке современных программных и аппаратных средств исследования и проектирования, контроля, технического диагностирования и промышленных испытаний систем автоматического и автоматизированного управления

1.1. Визуализация данных

Для начала посмотрим что из себя представляет наша база и с какими данными нам предстоит работать (***листинг 1***). Из (рис. 1.1 – 1.2) можно сделать вывод, что база данных состоит из таких переменных, как "name" (название машины), "mpg" (потребление топлива), "cylinders" (количество цилиндров), "displacement" (объем двигателя), "horsepower" (мощность двигателя), "weight" (вес автомобиля), "acceleration" (разгон), "model_year" (год выпуска) и "origin" (происхождение автомобиля), а также какой тип данных используется у каждой из переменных.

листинг 1

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.font_manager as fm
df = pd.read_csv(r"F:/Automobile.csv")  #добавляем базу
print(df.head(10))  #просмотр базы данных
df.info()   #информация о базе данных
```

```
                    name   mpg  cylinders  displacement  horsepower  weight  acceleration  model_year origin
0  chevrolet chevelle malibu  18.0          8         307.0       130.0    3504          12.0          70    usa
1          buick skylark 320  15.0          8         350.0       165.0    3693          11.5          70    usa
2         plymouth satellite  18.0          8         318.0       150.0    3436          11.0          70    usa
3              amc rebel sst  16.0          8         304.0       150.0    3433          12.0          70    usa
4                ford torino  17.0          8         302.0       140.0    3449          10.5          70    usa
5           ford galaxie 500  15.0          8         429.0       198.0    4341          10.0          70    usa
6           chevrolet impala  14.0          8         454.0       220.0    4354           9.0          70    usa
7          plymouth fury iii  14.0          8         440.0       215.0    4312           8.5          70    usa
8           pontiac catalina  14.0          8         455.0       225.0    4425          10.0          70    usa
9         amc ambassador dpl  15.0          8         390.0       190.0    3850           8.5          70    usa
```

Рисунок 2.1.1. Первые 10 строчек базы данных.

```
RangeIndex: 398 entries, 0 to 397
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   name          398 non-null    object
 1   mpg           398 non-null    float64
 2   cylinders     398 non-null    int64
 3   displacement  398 non-null    float64
 4   horsepower    392 non-null    float64
 5   weight        398 non-null    int64
 6   acceleration  398 non-null    float64
 7   model_year    398 non-null    int64
 8   origin        398 non-null    object
dtypes: float64(4), int64(3), object(2)
memory usage: 28.1+ KB
```

Рисунок 2.1.2. Информация о том, из чего состоит база данных.

Посмотрим какие марки автомобилей наиболее популярны, для этого мы очистим столбец 'name', чтобы оставить только марку автомобиля, группируем данные по столбцам 'name' и 'origin' и подсчитывает количество автомобилей каждой марки и происхождения (***листинг 2***).

листинг 2

```
#Топ 20 наиболее популярных марок машин
def top_20():
    #удаляем всё, что после пробела и остаётся только марка автомобиля
    df['name'] = df['name'].apply(lambda x: x.split(' ')[0])
    #создание таблицы name, origin с подсчётом количеста машин
    model_origin = df.groupby(['name','origin'],
as_index=False).size().sort_values(by='size', ascending=False)
    plt.figure(figsize=(10,6)) #размеры окна вывода
    ax = sns.barplot(x = model_origin.iloc[0:20, 2], y =
model_origin.iloc[0:20, 0])

    plt.title("Топ 20 наиболее популярных марок машин",fontsize=20)
    plt.xlabel('Колличество машин',fontsize=15)
    plt.ylabel('Марки машин',fontsize=15)

    labels = model_origin['origin'].values
    #добавляем страны сбоку
    for rect, label in zip(ax.patches, labels):
       width = rect.get_width()
       ax.text(width, rect.get_y() + rect.get_height() / 2,  label,
ha="left",    va="center",fontsize=10)
    plt.show()
top_20()
```

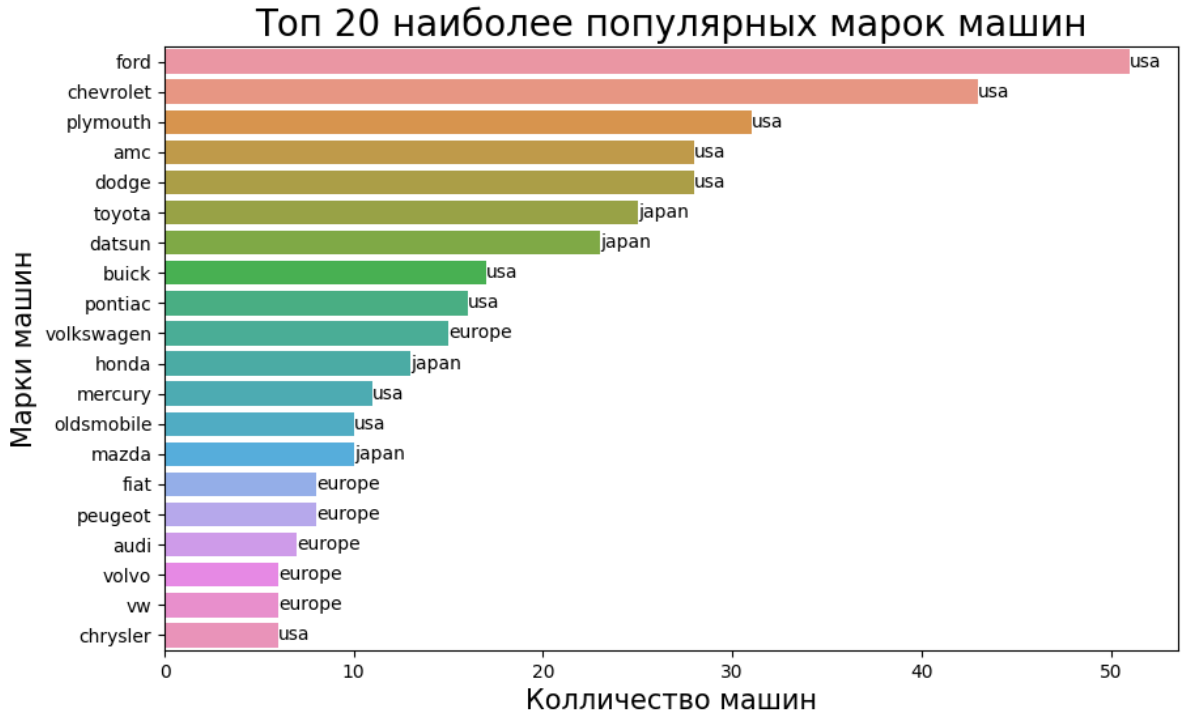


Рисунок 2.1.3. Самые распространённые марки автомобиля.

Посмотрим, какая страна преобладает в машиностроении. Для этого нам нужно сгруппировать данные по столбцу 'origin' и подсчитать количество автомобилей в каждой стране (***листинг 3***). А также создадим два графика: первый - столбчатая диаграмма, отображающая количество автомобилей в каждой стране, второй - круговая диаграмма, отображающая процентное соотношение количества автомобилей в каждой стране (рис. 1.3).

листинг 3

```
#Определяем кол-во машин в странах
def origin_c():
    #подсчёт кол-ва машин по странам
    origin_count = df.groupby('origin').size().sort_values()
    origin_count = origin_count.rename('count').reset_index()

    #создаём 2 графика
    fig, (ax1, ax2) = plt.subplots(nrows = 1, ncols = 2, figsize =
(10, 6))
    sns.barplot(x = 'origin', y = 'count', data = origin_count, ax =
ax1)

    ax1.set_title("Автомобилестроение в разных странах(кол-во)")
    ax1.set(xlabel='Страны' ,ylabel = 'Колличество машин')
    ax1.bar_label(ax1.containers[0])

    ax2.pie(origin_count['count'], labels = origin_count['origin'],
autopct = '%1.1f%%')
    ax2.set_title("Автомобилестроение  в разных странах(%)")
    plt.tight_layout()

    plt.show()
```

```
origin_c()
```

Рисунок 2.1.4. Столбчатая диаграмма и круговая диаграмма, показывающая автомобилестроение в разных странах.

Визуализируем переменные: "mpg" (потребление топлива), "cylinders" (количество цилиндров), "displacement" (объем двигателя), "horsepower" (мощность двигателя), "weight" (вес автомобиля), "acceleration" (разгон), "model_year" (год выпуска) (рис. 1.5. – 1.6.). Для визуализации создадим 3 графика: первый - блок-схема, отображающая разброс значений указанного признака (feature) в датафрейме df с помощью ящика с усами (boxplot); второй - гистограмма, отображающая распределение значений указанного признака (feature) в датафрейме df. График также включает ядерную оценку плотности (kde); и третий - столбчатая диаграмма, отображающая среднее значение указанного признака (feature) для каждой марки автомобиля (***листинг 4***).

листинг 4

```
#график распределения и прямоугольный график
def three_plots_num_column(feature):
    #удаляем всё, что после пробела и остаётся только марка автомобиля
    df['name'] = df['name'].apply(lambda x: x.split(' ')[0])
    plt.figure(figsize = (10,8))

    t1 = np.arange(0.0, 3.0, 0.01)
    ax1 =  plt.subplot(222)
    ax1.margins(0.05)
    ax1.set_title('Гистограмма',fontsize = 20)
    sns.histplot(data = df, x = feature, kde = True)

    ax2 = plt.subplot(221)
    ax2.margins(0.1)
    ax2.set_title('Блок-схема',fontsize = 20)
    sns.boxplot(y = df[feature])

    #группировка
    car_by_feature = df.groupby('name',as_index =
False)[feature].mean().sort_values(
    by = feature,
    ascending = False)
    #построение
    ax3 = plt.subplot(212)
    ax3.margins(0.05)
    ax3.set_title(f'По маркам ({feature}) ', fontsize = 18);
    sns.barplot(x = car_by_feature.iloc[0:20, 1], y =
car_by_feature.iloc[0:20, 0])

    plt.show()
three_plots_num_column('model_year')
```

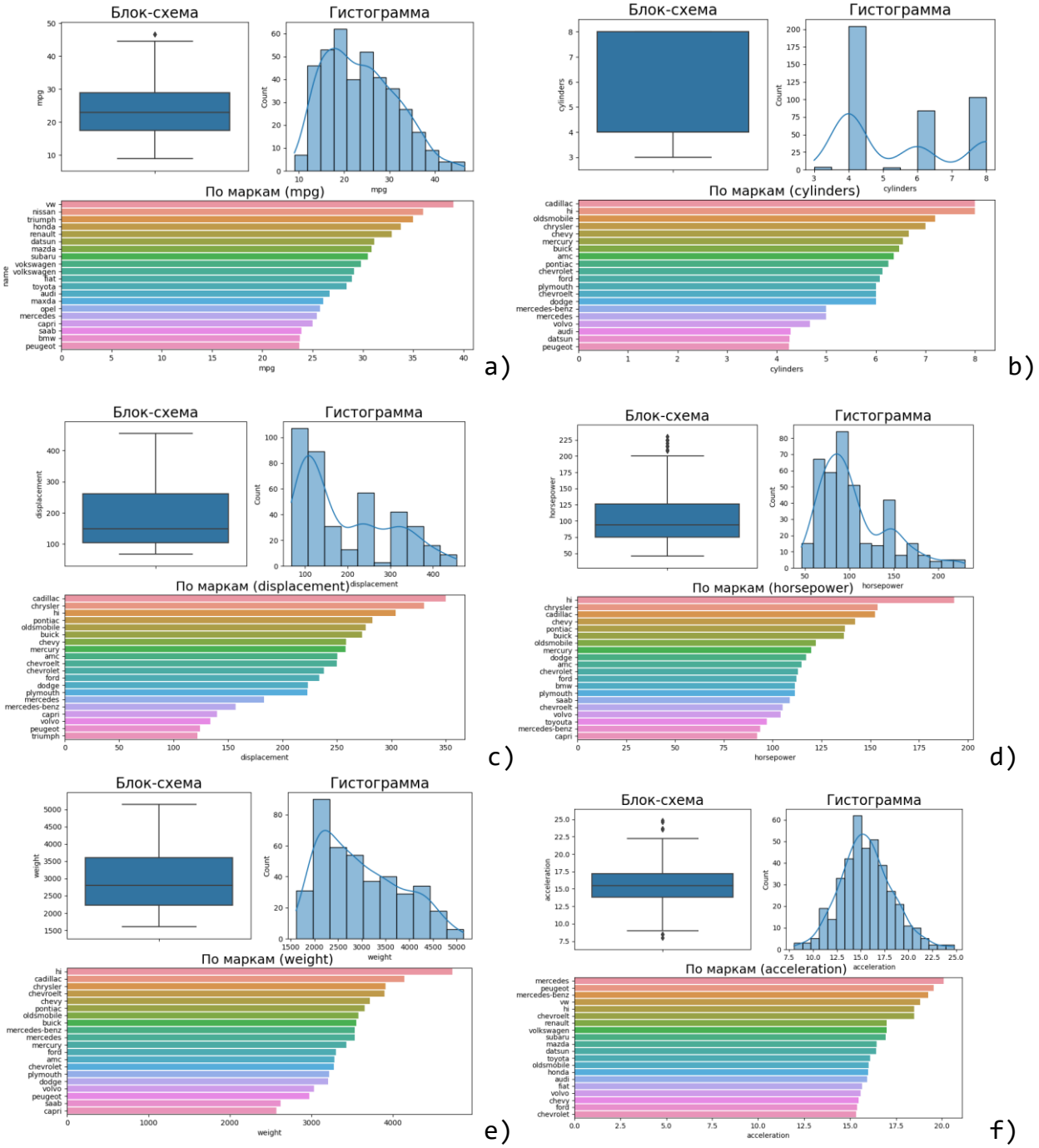
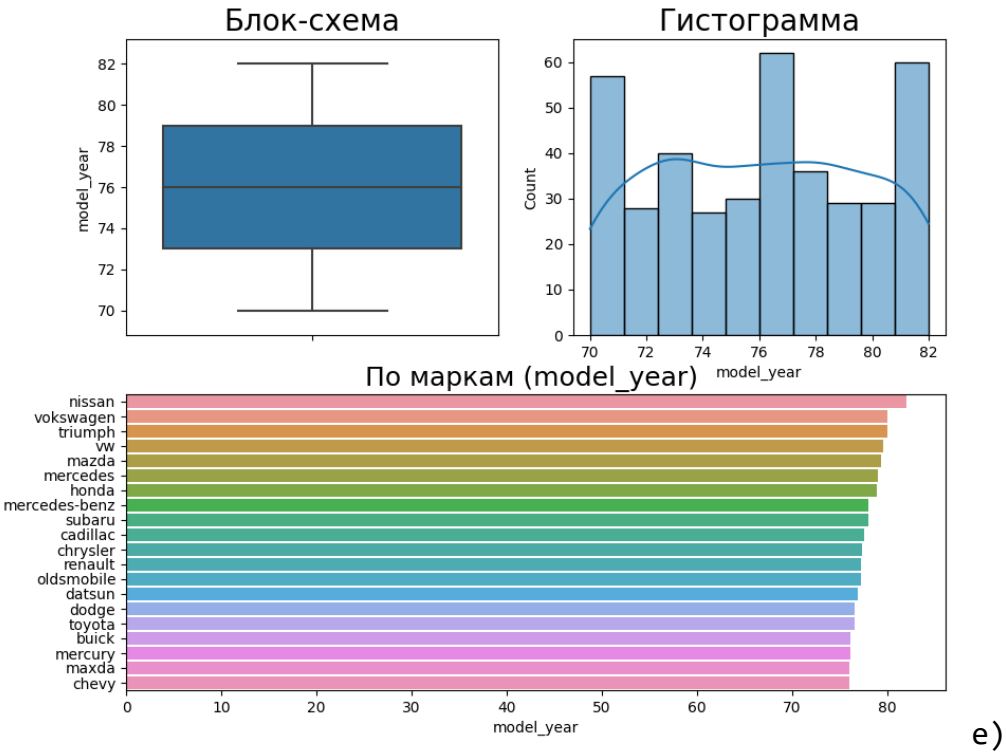


Рисунок 2.1.5. Визуализация данных (a – для "mpg"; b – для "cylinders"; c – для "displacement"; d – для "horsepower"; e – для "weight"; f – для "acceleration").



е)

Рисунок 2.1.6. Визуализация данных для "model_year".

1.2. Машинное обучение и поиск лучшего автомобиля

Проведём машинное обучение и выберем автомобиль (***листинг 5***). Для этого мы сначала запросим у пользователя приоритетные параметры, потом программа на основе этих данных проведёт обучение различными методами, просчитает точность каждого метода и выведет самый подходящий автомобиль (рис. 2.1).

листинг 5

```
#библиотеки для прогнозирование различными методами
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.linear_model import Ridge
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
#прогнозирование различными методами и выбор лучшего автомобиля

def models():
     df = pd.read_csv(r"F:/Automobile.csv")
     df = df.fillna(0) #где-то в базе отсутствует значение, чтобы не
было ошибки, ставим 0
     print("Параметры: mpg, cylinders, displacement, horsepower,
weight, acceleration, model_year")
     k_y = input("Введите параметр, который будет приоритетным при
выборе машины: ")
     k_x = [k_y,'name','origin']
```

```
    end = False
    ch = 0 #необходим, чтобы завершить добавление приоритетов

    while end != True and ch != 3:
        if int(input("Хотите добавить ещё приоритетный параметр?(да - 1; нет - 0): ")) == 1:
            k_x.append(input("Какой?: "))
        else:
            end = True
        ch += 1
    x = df.drop(k_x,axis = 1)
    y = df[k_y]
    nn = ['Линейная регрессия','Регрессия лассо','Ридж-регрессия','Регрессия дерева решений','Случайная регрессия леса','Классификация K-ближайших соседей']
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
    j = 0
    mas = []
    h = 0
    print("\n"+"======"*10)
    for i in nn:
        j += 1
        if j == 1:
            model = LinearRegression()
        elif j == 2:
            model = Lasso()
        elif j == 3:
            model = Ridge()
        elif j == 4:
            model = DecisionTreeRegressor()
        elif j == 5:
            model = RandomForestRegressor()
        elif j == 6:
            model = KNeighborsRegressor()
        #модель "линейная регрессия"
        model.fit(x_train, y_train)
        #оцениваем точность методов
        print(i + '(точность): ' + str(model.score(x_test, y_test)))
        if model.score(x_test, y_test) > h:
            h = model.score(x_test, y_test)
            k = model
            ki = i

    #--------
    print(("\n"+"======"*10)*2)
    print("Лучший метод: " + ki)
    model = k
    model.fit(x_train, y_train)
    predictions = model.predict(x_train)
    best_car_index = predictions.argmax()
```

```
    best_car = df.iloc[best_car_index]
    print("Самая лучшая машина по результатам всех методов:")
    print(best_car)

models()
Параметры: mpg, cylinders, displacement, horsepower, weight, acceleration, model
_year
Введите параметр, который будет приоритетным при выборе машины: mpg
Хотите добавить ещё приоритетный параметр?(да - 1; нет - 0): 1
Какой?: cylinders
Хотите добавить ещё приоритетный параметр?(да - 1; нет - 0): 0

============================================================
Линейная регрессия(точность): 0.8254253208629132
Регрессия лассо(точность): 0.8307230756346722
Ридж-регрессия(точность): 0.8254462435012726
Регрессия дерева решений(точность): 0.8237771677060193
Случайная регрессия леса(точность): 0.9025940731387874
Классификация К-ближайших соседей(точность): 0.7642398470886407

============================================================
============================================================
Лучший метод: Случайная регрессия леса
Самая лучшая машина по результатам всех методов:
name            volkswagen rabbit custom
mpg                               29.0
cylinders                            4
displacement                      97.0
horsepower                        78.0
weight                            1940
acceleration                      14.5
model_year                          77
origin                          europe
Name: 233, dtype: object
```

Рисунок 2.2.1. Выбор приоритетных параметров, вычисление точности методов и нахождение машины на основе лучшего метода.

Заключение

Во время прохождения практики были реализованы следующие задачи:

1. Ознакомиться с историей и приоритетными направлениями деятельности производства предприятия АО «Тайфун».

2. Изучена теория машинного обучения.

3. Изучено несколько методов машинного обучения, а именно: линейная регрессия, случайный лес, регрессия лассо, ридж-регрессия, регрессия дерева решений и метод k ближайших соседей

4. Визуализация данных базы.

5. Разработано консольное приложение поиска наилучшего автомобиля.

6. Оформлен отчёт о проделанной работе.