

**Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное учреждение
высшего образования**

«Российский экономический университет имени Г. В. Плеханова»

Высшая школа кибертехнологий, математики и статистики

Кафедра информатики

ОТЧЁТ

по лабораторной работе

**на тему «Прогнозирование цен на ноутбуки на основе методов
машинного обучения»**

Выполнила:

Студентки 3 курса

группы 15.11Д-БИЦТ09/216

Крабу Кира Сергеевна

Преподаватель:

к.т.н. Ярушев Сергей Александрович

Москва

2023

Оглавление

1. Бизнес-анализ.	3
2. Анализ данных.	4
3. Подготовка данных	11
4. Моделирование.....	12
5. Оценка результатов.....	14
6. Внедрение.	14
Приложение.....	15

1. Бизнес-анализ.

Цель проекта – построить модель определения цены в зависимости от технических характеристик ноутбука, которая позволит прогнозировать его рыночную стоимость. Рынок ноутбуков имеет большое разнообразие технических характеристик, основных производителей, моделей, факторов, которые могут влиять на цены на ноутбуки. Модель по прогнозированию цен на ноутбуки в зависимости от конкретно выбранных параметров может быть использована для эффективной оценки необходимых ресурсов, определения ценовой политики производителем, увеличения объема продаж и улучшения уровня удовлетворенности своих клиентов.

Риски проекта:

1. Несоблюдение сроков проекта;
2. Риск неплатежеспособности заказчика;
3. Риск нехватки и неполноты данных;
4. Риск несоответствия полученных результатов требованиям заказчика.

Метрики оценки точности и качества построенных моделей:

Для моделей регрессии: качество модели определяется с использованием коэффициента детерминации (R^2), точность модели определяется на основании средней относительной ошибки (MAPE). Границы значений метрик: R^2 должен быть больше либо равен 0.8, MAPE не более 10%.

Ожидаемые преимущества: увеличение прибыли компании по производству ноутбуков на 5% в среднесрочной перспективе. Критерий успеха – рост объёма продаж за счёт лучшего понимания необходимых финансовых ресурсов для производства ноутбука и потребностей и возможностей клиентов.

Задачи анализа данных, решаемые в рамках проекта:

1. Построение визуализации ценового диапазона ноутбуков на рынке, ведущих “игроков” рынка ноутбуков, средних ценовых

предложений конкурентов, сравнительного анализа технических характеристик.

2. Решение задачи прогнозирования цены на единицу продукции с использованием моделей регрессии. В качестве моделей регрессии будут рассмотрены методы: линия тренда, метод ближайшего соседа, метод дерева решений и метод случайного леса.

2. Анализ данных.

Данные, которые будут анализироваться:

- id ноутбука;
- Компания-производитель;
- Наименование товара;
- Тип ноутбука;
- Диагональ экрана;
- Разрешение экрана;
- CPU;
- RAM;
- Память;
- GPU;
- Операционная систем;
- Вес ноутбука;
- Цена в евро.

(<https://www.kaggle.com/datasets/muhammetvarl/laptop-price>)

Объем данных – 193 Кбайт.

Формат данных – файл csv, разделитель – “,”.

Price_euros – прогнозируемая, выходная переменная.

Таблица 1. Типы, виды данных и схемы кодирования.

Наименование	Тип данных	Вид данных	Схема кодирования
laptop_ID	Целое число	Непрерывный	-
Company	Строковый	Дискретный	Целое число
Product	Строковый	Дискретный	Целое число
TypeName	Строковый	Дискретный	Целое число
Inches	Число с плавающей запятой	Непрерывный	-
ScreenResolution	Строковый	Дискретный	Целое число
Cpu	Строковый	Дискретный	Целое число
Ram	Строковый	Дискретный	Целое число
Memory	Строковый	Дискретный	Целое число
Gpu	Строковый	Дискретный	Целое число
OpSys	Строковый	Дискретный	Целое число
Weight	Строковый	Дискретный	Целое число
Price_euros	Число с плавающей запятой	Непрерывный	-

Начнем писать код на Python. Импортируем нужные нам библиотеки.

Pandas – это библиотека для работы с панельными данными (с таблицами). Понадобится нам для представления входных данных.

Matplotlib и seaborn – библиотеки визуализации.

NumPy – для работы с массивами.

Sklearn – библиотека для создания линейной модели, тестовых и обучающих выборок, для метрик качества модели.

```
import pandas as pd
```

```
import matplotlib.pyplot as plt
```

```

import numpy as np
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score, mean_absolute_percentage_error

```

Функция Encode нужна нам для того, чтобы упростить последующую работу с данными, унифицировать значения в датасете. Переменным в строчном виде присваивается уникальный номер, чтобы придать им числовой вид.

```

def Encode (df):
    for column in df.columns:
        if df[column].dtype != 'int64' and df[column].dtype != 'float64':
            enc = LabelEncoder()
            enc.fit(df[column])
            df[column] = enc.transform(df[column])
    return df

```

Проанализируем количество пропусков и дубликатов в наших данных. Далее они будут удалены для того, чтобы информация была более валидна.

```

df = pd.read_csv('laptop_price.csv', encoding= 'cp1251')
print (df.info())
df.drop('laptop_ID', inplace = True, axis = 1)
print (df.describe())
df = df.drop_duplicates ()
print (df.info())

```

Проверка пропущенных значений:

laptop_ID	0
Company	0
Product	0
TypeName	0
Inches	0
ScreenResolution	0
Cpu	0
Ram	0
Memory	0
Gpu	0
OpSys	0
Weight	0
Price_euros	0

Пропущенные значения отсутствуют.

Проведем анализ на качество данных. Функция `print (df.describe())` выводит нам таблицу описательной статистики. Чтобы более наглядно представить некоторые данные, выведем:

1. Рисунок 1, цены по ноутбукам:

```
plt.figure(figsize = (7, 7))
y_prices = np.array(df['Price_euros'])
x_id = np.array(range(len(df)))
plt.scatter (x_id, y_prices, color = 'red')
plt.title ('Цены ноутбуков')
plt.xlabel ('laptop')
plt.ylabel ('Price')
plt.savefig('Lap prices.png')
plt.show()
```

Также на рисунке видны выбросы в ценовом факторе.

2. Рисунок 2, сколько ноутбуков у производителя:

```
plt.figure(figsize = (30, 12))
plt.subplot(1, 2, 1)
counts = df['Company'].value_counts(sort = False)
plt.title ('Компании и их доли в датасете', fontdict = {'fontsize' : 30})
plt.pie(counts, labels = brands, labeldistance=0.7, radius=1.2,
textprops={'fontsize': 24})
```

3. Рисунок 2, средние цены по производителям:

```
brand_mean_price = []
for i in range(len(brands)):
    brand_mean_price.append(df[df['Company'] == i].Price_euros.mean())
plt.subplot(1,2,2)
plt.bar(brands, brand_mean_price, color = 'black')
plt.xlabel('Companies', fontdict = {'fontsize' : 20})
plt.ylabel('Mean Price', fontdict = {'fontsize' : 20})
plt.title ('Средняя цена за ноутбук у компаний', fontdict = {'fontsize' : 30})
plt.savefig('Price factors.png')
plt.show()
plt.close('all')
```

Рассмотрим визуализацию изначальных данных. На рисунке 1 видны точечные выбросы по ценам некоторых ноутбуков, с которым впоследствии будем работать с помощью метода трёх сигм.

На рисунке 2 можно увидеть основных лидеров рынка по производству ноутбуков, а также средние цены по имеющимся данным тех или иных брендов.

Как уже было замечено ранее, данные «Price euros» имеют заметные экстремальные значения, которые в дальнейшем могут повлиять на качество модели: особенно это видно из разницы между статистиками 75-персентиля и максимального значения. Их обработаем методом «3 сигм», то есть те значения, которые отклоняются от среднего на 3 стандартных отклонения, являются нежелательными для нас выбросами, которые следует исключить для дальнейшего анализа.

Функция для этого: $df = df[df['Price_euros'] < 3*df['Price_euros'].std() + df['Price_euros'].mean()]$
`print (df.describe())`

Так же в результате исполнения кода была выведена матрица корреляций.

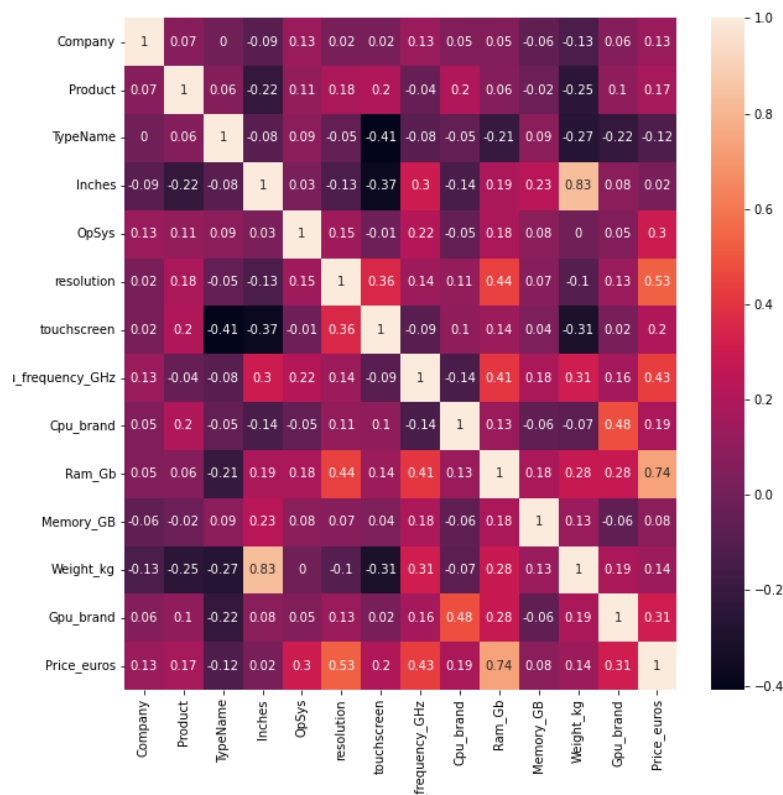


Рисунок 3. – Корреляционная матрица

Теперь мы можем сделать следующие выводы:

1. Данные не качественные. В данных представлены выбросы.
2. В данных присутствует линейная зависимость цены от RAM (оперативной памяти) и разрешения экрана. Однако, оперативная память

и разрешение экрана тоже коррелируют (мультиколлинеарность), поэтому мы можем взять только один из них.

3. Присутствует корреляция между ценой ноутбука и частотой процессора. Однако не существенная.

3. Подготовка данных

Данные требуют значительной подготовки, очистки. Выявлено 28 дубликатов, 12 выбросов, 0 пропусков. Переменная-индикатор 'laptop_ID' была удалена.

Также многие категории данных переделаны в более содержательный вид из соображений удобства работы с числовыми данными. Из переменной 'ScreenResolution' были сформированы новые столбцы: 'resolution' со значениями типа " $*x*$ " и 'touchscreen' как показатель применения технологии тачскрин в ноутбуке. Из переменной 'Cpu' было выделено значение частоты центрального процессора в Гц. Также взяли из переменных 'Cpu' и 'Gpu' отдельно производителя для дальнейшего анализа связи. 'Ram', 'Weight' и 'Memory' были изменены до числового вида без добавления "GB", "kg".

Таким образом, переменные 'ScreenResolution', 'Cpu', 'Ram', 'Memory', 'Gpu', 'Weight' были удалены. На их место пришли 'resolution', 'touchscreen', 'cpu_frequency_GHz', 'Cpu_brand', 'Ram_Gb', 'Memory_GB', 'Weight_kg', 'Gpu_brand'.

Оставшиеся строковые переменные получили новый уникальный номер благодаря функции Encode с целью придания им числового вида для дальнейшего анализа на основе машинного обучения.

После работы с факторами, оказывающими влияние на цену ноутбука, рассмотрим Рисунок 4 с графической связью между факторными переменными и нашей целевой.



Рисунок 4. – Связь факторов с целевой переменной.

4. Моделирование

Исходные данные имеют вид:

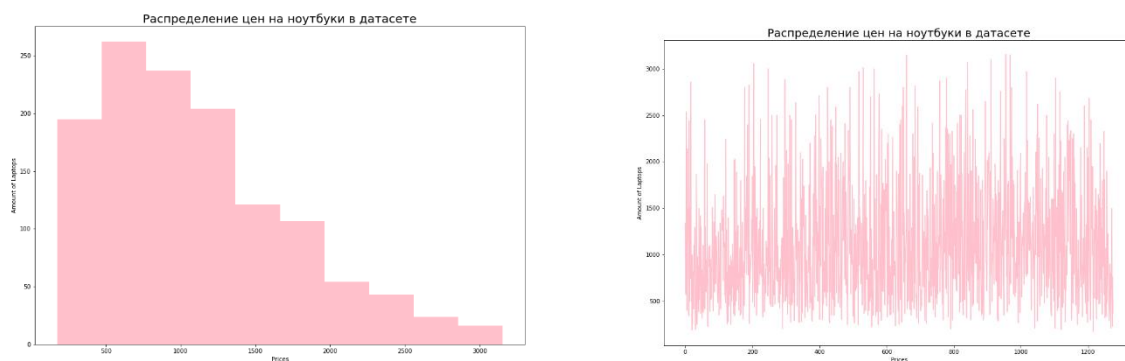


Рисунок 5. – Распределение цен на ноутбуки в исходном датасете.

В рамках данного проекта проводим моделирование с использованием линейной регрессии. Однако если включать в модели только те факторы, которые имеют высокую корреляцию с целевой переменной, то метрики модели выходят нехорошие. Самая оптимальная по оценке модель линейной регрессии та, которая содержит все факторы.

Было получено уравнение вида:

$$177.2442104446916 + 7.39516837 x_1 + 0.112133831 x_2 + 47.758959 x_3 - 107.265276 x_4 + 76.1875763 x_5 + 44.678216 x_6 + 129.099489 x_7 + 240.380273 x_8 + 198.803695 x_9 + 227.759098 x_{10} - 12.0590746 x_{11} + 218.855082 x_{12} + 46.0839755 x_{13} = y$$

y – выходная переменная (спрогнозированная цена)

коэффициенты регрессии для факторов:

x_1 – компания-производитель

x_2 – название продукта

x_3 – тип ноутбука

x_4 – диагональ экрана

x_5 – операционная система

x_6 – разрешение

x_7 – наличие тачскрина

x_8 – частота центрального процессора

x_9 – производитель центрального процессора

x_{10} – размер оперативной памяти

x_{11} – размер постоянной памяти

x_{12} – производитель графического процессора

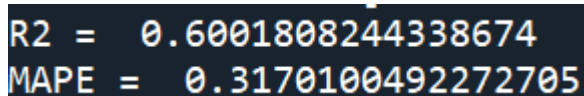
x_{13} – частота центрального процесса

Выведем непосредственно результаты модели регрессии, а именно коэффициент детерминации и ошибку MAPE:

```
def lin_reg(x_train, x_test, y_train, y_test):  
    reg = LinearRegression()  
    print ('_____Linear_____')
```

```
reg.fit(x_train, y_train)
y_pred = reg.predict(x_test)
print (reg.intercept_, reg.coef_)
print ('R2 = ', r2_score(y_test, y_pred))
print ('MAPE = ', mean_absolute_percentage_error(y_test, y_pred))
return y_pred
```

5. Оценка результатов

A screenshot of a terminal window showing the results of a regression model. The text is displayed in a monospaced font with a light blue background. The first line shows 'R2 = 0.6001808244338674' and the second line shows 'MAPE = 0.3170100492272705'.

```
R2 = 0.6001808244338674
MAPE = 0.3170100492272705
```

Рисунок 6. – Коэффициент детерминации и ошибка MAPE

Значение коэффициента детерминации является неприемлемым и делает полученную модель не оправдавшей наших результатов, а также средняя относительная ошибка превышает поставленный уровень на 20 процентов. Принятие модели в эксплуатацию зависит от заказчика. Очевидно, что модель будет нуждаться в доработке.

В качестве рекомендаций следует сказать, что предполагаемая благоприятная сфера для внедрения данной модели – это ценовая политика компании по производству ноутбуков. Модель при лучших результатах могла бы позволить на основе анализа уже существующих цен и спроса на ноутбуки определять комфортную ценовую категорию для потребителей ноутбуков.

6. Внедрение.

Доработанная регрессионная модель позволит компании более точно планировать свои доходы и расходы, установить оптимальные цены на ноутбуки и более эффективно конкурировать на рынке. Также, использование модели для прогнозирования цен может помочь предприятию увеличить объем продаж и улучшить уровень удовлетворенности своих клиентов.

В качестве потенциальных путей внедрения модели следует возможность оптимизации стоимости производства ноутбуков заказчика-производителя: благодаря знанию значимости факторов цены ноутбука будет легче не упускать потребительский спрос наряду с уменьшением издержек.

Также усовершенствованное ценообразование в компании может стать хорошей маркетинговой стратегии с последующим ориентиром на финансовые возможности своей целевой аудитории.

Приложение

Приложение 1. Код программы Python для проведенного анализа.

```
# -*- coding: utf-8 -*-  
# Прогнозирование цен на ноутбуки на основе методов машинного обучения  
  
import pandas as pd  
import matplotlib.pyplot as plt  
import numpy as np  
import seaborn as sns  
from sklearn.preprocessing import LabelEncoder  
from sklearn.linear_model import LinearRegression  
from sklearn.model_selection import train_test_split  
from sklearn.metrics import r2_score, mean_absolute_percentage_error  
  
def Encode (df):  
    for column in df.columns:  
        if df[column].dtype != 'int64' and df[column].dtype != 'float64':  
            enc = LabelEncoder()  
            enc.fit(df[column])  
            df[column] = enc.transform(df[column])  
    return df
```

```
def lin_reg(x_train, x_test, y_train, y_test):
    reg = LinearRegression()
    print ('_____Linear_____')
    reg.fit(x_train, y_train)
    y_pred = reg.predict(x_test)
    print (reg.intercept_, reg.coef_)
    print ('R2 = ', r2_score(y_test, y_pred))
    print ('MAPE = ', mean_absolute_percentage_error(y_test, y_pred))
    return y_pred
```

```
df = df.drop_duplicates ()
print (df.info())
```

далее написаны функции для того, чтобы входящие характеристики
разделились на точные данные

```
print (df['ScreenResolution'].unique())
df['resolution'] = df['ScreenResolution'].str.extract(pat = '(\d+x\d+)')
df['touchscreen'] = df['ScreenResolution'].str.extract(pat = '(Touchscreen)')
df['touchscreen'] = df['touchscreen'].replace('Touchscreen',1)
df['touchscreen'] = df['touchscreen'].replace(np.nan,0)
df.drop('ScreenResolution', inplace = True, axis = 1)
df['touchscreen'] = df['touchscreen'].astype(int)
```

```
print (df.Cpu.unique())
df['Cpu'] = df['Cpu'].str.split()
print (df['Cpu'])
df['Cpu1'] = [i[-1] for i in df['Cpu']]
```



```
df['cpu_frequency_GHz'] = df['Cpu1'].apply(lambda i: str(i[:-3]))
df['cpu_frequency_GHz'] = df['cpu_frequency_GHz'].astype(float)
df['Cpu_brand'] = [i[0] for i in df['Cpu']]
df.drop(['Cpu', 'Cpu1'], inplace = True, axis = 1)
print (df['Cpu_brand'].unique())
```

```
print (df.Ram.unique())
df['Ram_Gb'] = df['Ram'].apply(lambda i: str(i[:-2]))
df.drop('Ram', inplace = True, axis = 1)
df['Ram_Gb'] = df['Ram_Gb'].astype(int)
```

```
print (df.Memory.unique())
df['Memory']=df['Memory'].str.replace('1.0TB','1TB', regex=True)
df['Memory']=df['Memory'].str.replace('1TB','1000GB')
df['Memory']=df['Memory'].str.replace('2TB','2000GB')
df['Memory'] = df['Memory'].str.split()
df['Memory_GB'] = [i[0] for i in df['Memory']]
df['Memory_GB'] = df['Memory_GB'].apply(lambda i: str(i[:-2]))
df['Memory_GB'] = df['Memory_GB'].astype(int)
df['Memory_GB'] = df['Memory_GB'].astype(int)
df.drop('Memory', inplace = True, axis = 1)
```

```
print (df.Weight.unique())
df['Weight_kg'] = df['Weight'].apply(lambda i: str(i[:-2]))
df.drop('Weight', inplace = True, axis = 1)
df['Weight_kg'] = df['Weight_kg'].astype(float)
```

```
print (df.Gpu.unique())
df['Gpu_brand'] = df['Gpu'].str.split()
df['Gpu_brand'] = [i[0] for i in df['Gpu_brand']]
```

```
df.drop('Gpu', inplace = True, axis = 1)
```

```
cols = list(df.columns)
```

```
cols = cols[0:5] + cols[6:] + [cols[5]]
```

```
df = df [cols]
```

```
brands = df['Company'].unique()
```

```
print (df.info())
```

```
df = Encode(df)
```

```
# анализ на качество данных
```

```
print (df.describe())
```

```
# цены по ноутбукам
```

```
plt.figure(figsize = (7, 7))
```

```
y_prices = np.array(df['Price_euros'])
```

```
x_id = np.array(range(len(df)))
```

```
plt.scatter (x_id, y_prices, color = 'red')
```

```
plt.title ('Цены ноутбуков')
```

```
plt.xlabel ('laptop')
```

```
plt.ylabel ('Price')
```

```
plt.savefig('Lap prices.png')
```

```
plt.show()
```

```
plt.close('all')
```

```
# видны выбросы в ценовом факторе
```

```
# сколько ноутбуков у производителя
```

```
plt.figure(figsize = (30, 12))
```

```
plt.subplot(1, 2, 1)
```

```

counts = df['Company'].value_counts(sort = False)
plt.title ('Компании и их доли в датасете', fontdict = {'fontsize' : 30})
plt.pie(counts, labels = brands, labeldistance=0.7, radius=1.2, textprops={'fontsize':
24})

# средние цены по производителям
brand_mean_price = []
for i in range(len(brands)):
    brand_mean_price.append(df[df['Company'] == i].Price_euros.mean())
plt.subplot(1,2,2)
plt.bar(brands, brand_mean_price, color = 'black')
plt.xlabel('Companies', fontdict = {'fontsize' : 20})
plt.ylabel('Mean Price', fontdict = {'fontsize' : 20})
plt.title ('Средняя цена за ноутбук у компаний', fontdict = {'fontsize' : 30})
plt.savefig('Price factors.png')
plt.show()
plt.close('all')

# метод 3 сигм
df = df[df['Price_euros'] < 3*df['Price_euros'].std() + df['Price_euros'].mean()]
print (df.describe())

# посмотрим графически степень влияния факторов
fig = plt.figure(figsize = (20, 20))
for i, item in enumerate(df.columns[0:-1], start = 1):
    graf = fig.add_subplot(5, 3, i)
    plt.scatter(df[item], df['Price_euros'])
    graf.set_title(item)
plt.savefig('Factors.png')
plt.show()
plt.close('all')

```

```

# корреляция
plt.figure(figsize = (10, 10))
sns.heatmap(df.corr())
sns.heatmap(df.corr().round(2), cbar = False, annot = True)
plt.savefig('corr_matrix.png')
plt.show()
plt.close('all')

# inches weight сильно коррелируют между собой
# на price влияет ram и cpu resolution

plt.figure(figsize = (15, 10))
plt.plot(df['Price_euros'], color = 'pink')
plt.xlabel('Prices', fontdict = {'fontsize' : 10})
plt.ylabel('Amount of Laptops', fontdict = {'fontsize' : 10})
plt.title('Распределение цен на ноутбуки в датасете', fontdict = {'fontsize' : 20})
plt.savefig('Prices.png')
plt.show()

df_factors = df[df.columns[:-1]]
x_train, x_test, y_train, y_test = train_test_split(df_factors, df[df.columns[-1]],
train_size = 0.8, random_state = 1)

y_pred = lin_reg(x_train, x_test, y_train, y_test)

```