Министерство образования и науки Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ
(НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)»
(МАИ)**
Институт №3 «Системы управления, информатика и электроэнергетика»
Кафедра №307 «Цифровые технологии и информационные системы»

Курсовая работа по дисциплине

"Анализ данных на Python"

Выполнила студентка
группы М3О-412Б-21 _____2024 г Подпись:_____
А.Р. Барабаш

Проверил д.ф-м.н.,
доцент НГТУ НЭТИ _____2024 г. Подпись:_____
А.П. Ковалевский

Москва 2024

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ

Разработать и оценить вероятностную модель на основе набора исходных данных, применяя изученные методы статистического анализа и моделирования для обеспечения точности модели и её предсказательной способности.

2 ЗАДАНИЕ

1. Найти и прочитать данные для исследования.
2. Построить вероятностную модель.
3. Исследовать адекватность модели статистическим тестом и/или стохастическим моделированием, кросс-валидацией.
4. На основании этого исследования исправить модель, а затем проверить адекватность исправленной модели.

3 ХОД РАБОТЫ

3.1 Поиск данных для исследования

В качестве исходного набора данных был выбран датасет pokemon.csv. Из этого датасета были выбраны и очищены от пропусков поля ’height_m’, ’weight_kg’, чтобы построить модель зависимости массы покемона от его роста.

```
df = pd.read_csv('pokemon.csv')

df = df.dropna(subset=['height_m', 'weight_kg'])

X = df[['height_m']]
y = df['weight_kg']
```

3.2 Построение математической модели

Для построения модели использовались библиотеки scikit-learn, результаты моделирования представлены на рисунках 1–3:

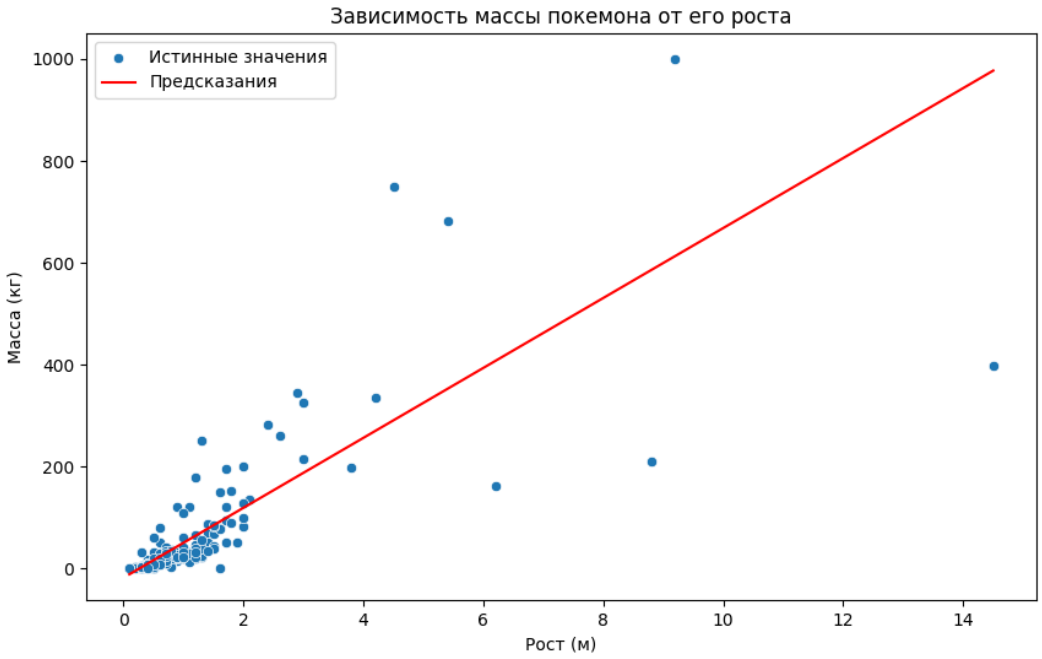

Рисунок 1: График зависимости массы покемона от его роста

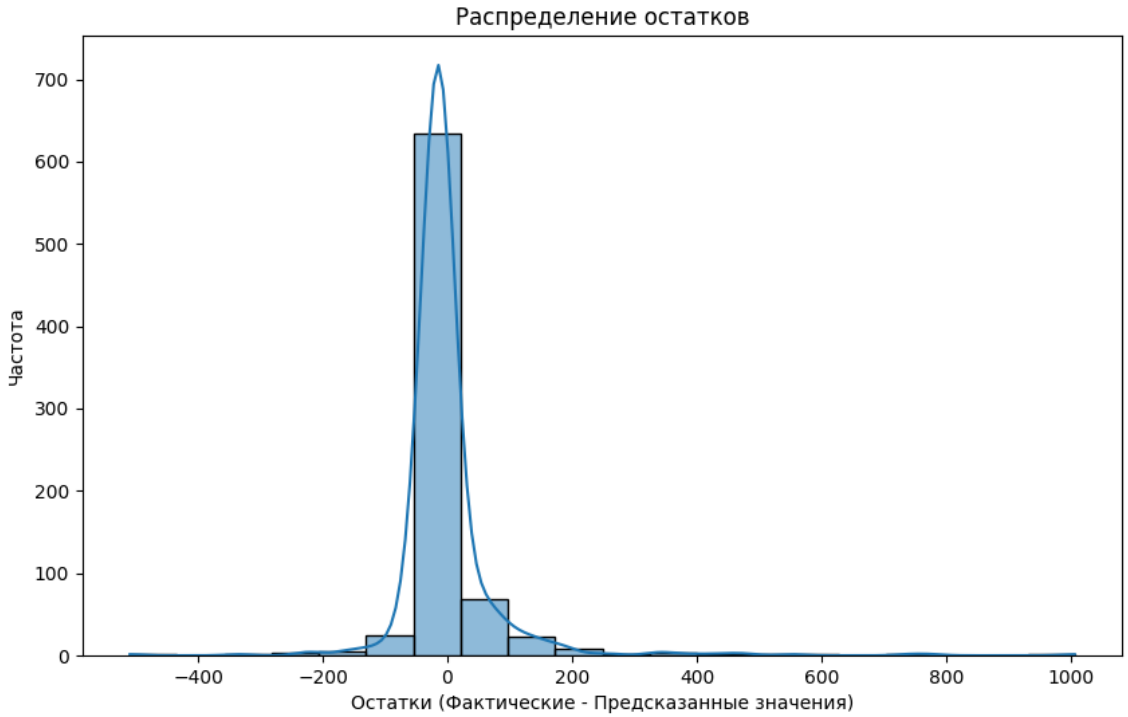

Рисунок 2: Распределение остатков

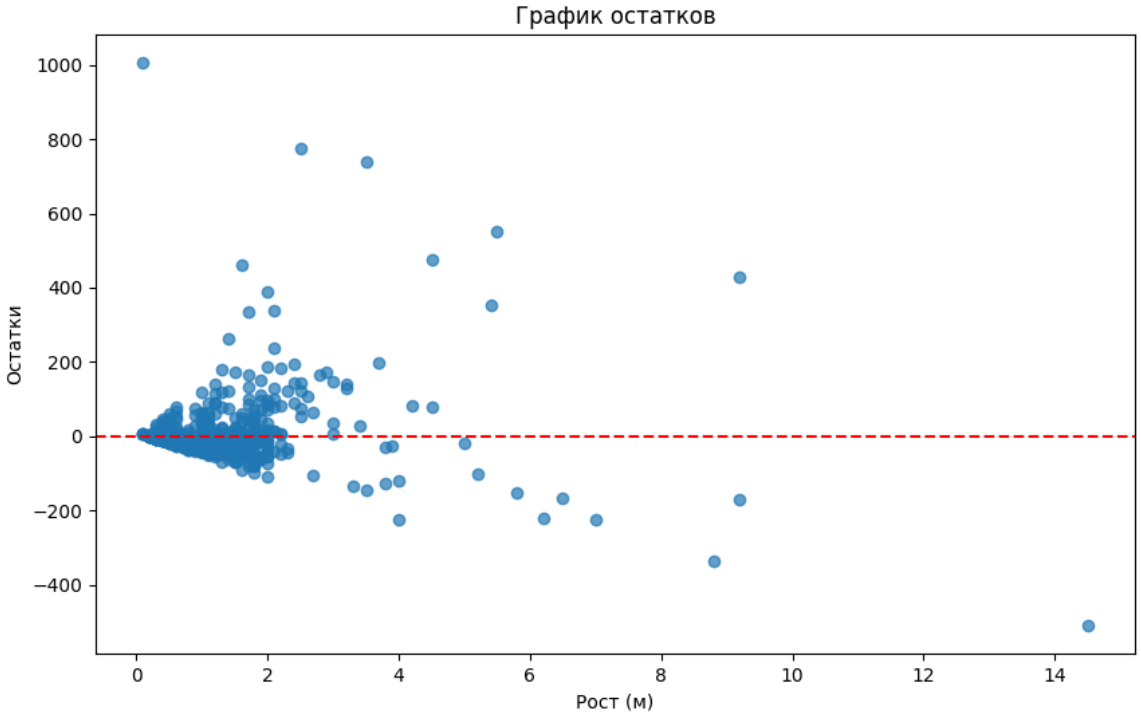

Рисунок 3: График остатков

Результаты аналитического изучения модели представлены в таблице 1:

Параметр	**Значение**
Коэффициент модели	68.6567
Свободный член	-18.28199
Коэффициент детерминации R^2 на тестовых данных	0.5260
Средний коэффициент детерминации R^2 по результатам кросс-валидации	0.0918
Стандартное отклонение R^2	0.0918
Статистика теста Шапиро-Уилка	0.5147
p-value	0.0000

Таблица 1: Результаты аналитического изучения модели

Коэффициент детерминации (R^2) на тестовых данных: 0.526. Этот результат означает, что модель объясняет примерно 52.6% дисперсии данных. Он указывает на то, что модель не учитывает всех факторов, влияющих на массу покемона.

Кросс-валидация: Средний R^2 по результатам кросс-валидации: 0.4012 и стандартное отклонение R^2: 0.0918. Эти значения показывают, что модель может давать несколько разные результаты в зависимости от подвыборки, что может говорить о нестабильности или наличии недостаточно объясняющих факторов.

Тест Шапиро-Уилка для остатков: p-значение: 0.0000 указывает на то, что остатки модели существенно отклоняются от нормального распределения. Это свидетельствует о проблемах с адекватностью модели: модель не описывает всю структуру данных, и её остатки содержат важную информацию, не объясненную текущей линейной зависимостью.

3.3 Улучшение математической модели

Попробуем прологарифмировать, очистить от выбросов и упорядочить по росту наши данные, чтобы получить более качественную математическую модель.

```
df = df.dropna(subset=['height_m', 'weight_kg'])
df = df.sort_values(by='height_m').reset_index(drop=True)
# Логарифмическое преобразование признаков (теперь после сортировки)
df['log_height_m'] = np.log(df['height_m'])
df['log_weight_kg'] = np.log(df['weight_kg'])
# Очистка от выбросов с использованием IQR (межквартильный размах)
def remove_outliers(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column_name] >= lower_bound) & (df[column_name] <= upper_bound)]

df = remove_outliers(df, 'log_height_m')
df = remove_outliers(df, 'log_weight_kg')
```

Проведём те же исследования, что и над исходными данными. Результаты представлены в рисунках 4-6 и таблице 2:

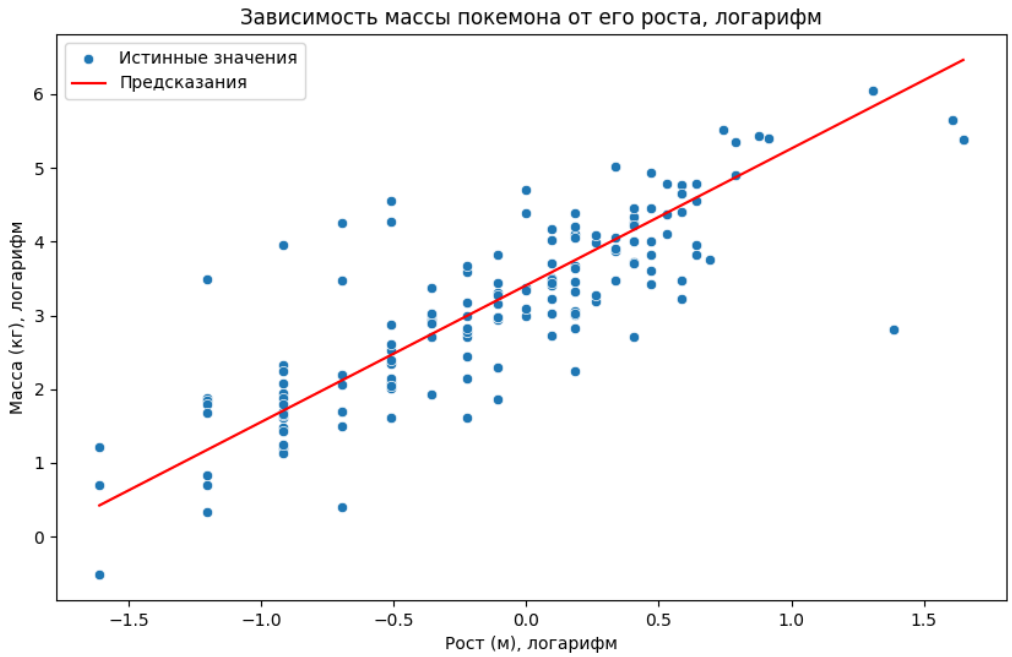

Рисунок 4: График зависимости массы покемона от его роста прологарифмированной модели

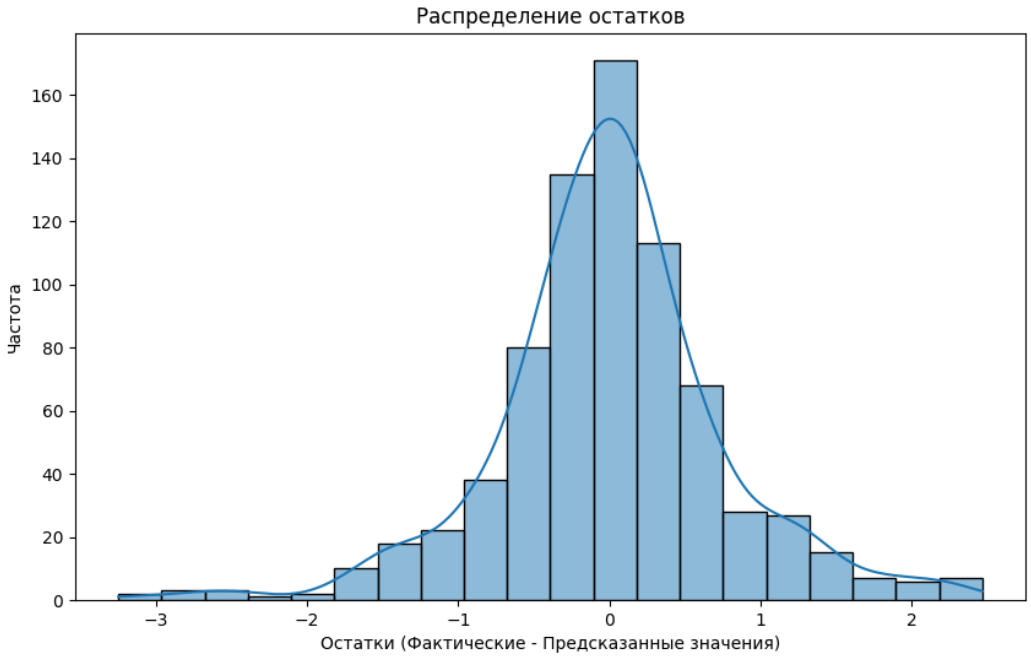

Рисунок 5: Распределение остатков прологарифмированной модели

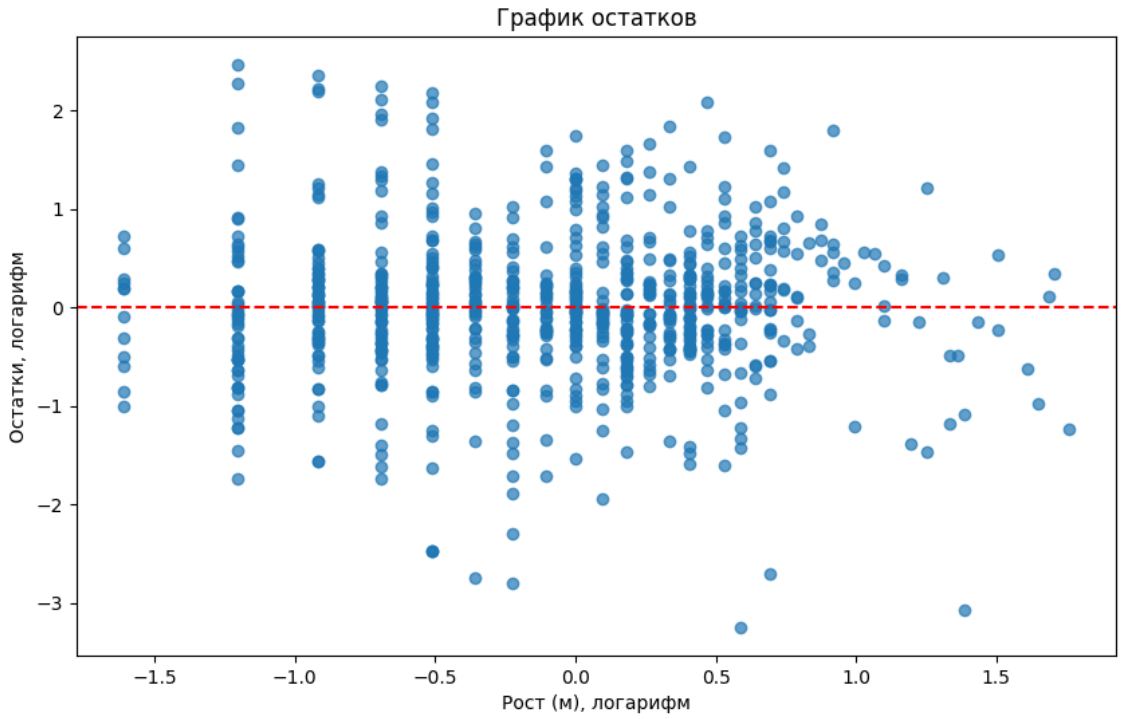

Рисунок 6: График остатков прологарифмированной модели

Результаты аналитического изучения модели представлены в таблице 2:

Параметр	**Значение**
Коэффициент модели	1.8529
Свободный член	3.4071
Коэффициент детерминации R^2 на тестовых данных	0.6399
Средний коэффициент детерминации R^2 по результатам кросс-валидации	0.6919
Стандартное отклонение R^2	0.0307
Статистика теста Шапиро-Уилка	0.9641
p-value	0.0000

Таблица 2: Результаты аналитического изучения прологарифмированной модели

Коэффициент модели: 1.8529 и свободный член (intercept): 3.4071 описывают линейную зависимость между логарифмом роста и логарифмом массы покемонов. Коэффициент модели означает, что при увеличении логарифма роста на единицу, логарифм массы увеличивается примерно на 1.85.

Коэффициент детерминации (R^2) на тестовых данных: 0.6399: это хорошее значение, которое показывает, что модель объясняет около 64% вариации в данных Это существенно лучше, чем R^2 для модели без логарифмирования (первоначально 0.526). Логарифмирование улучшило предсказательную способность модели и сделало зависимость более линейной.

Кросс-валидация: Средний R^2 по результатам кросс-валидации: 0.6919. Это значение подтверждает, что модель демонстрирует хорошую и стабильную предсказательную способность на разных подвыборках. Стандартное отклонение также уменьшилось, что говорит о большей стабильности модели.

Тест Шапиро-Уилка: Статистика теста: 0.9641 и p-значение: 0.0000. Хотя статистика теста Шапиро-Уилка несколько улучшилась по сравнению с первоначальной моделью (статистика увеличилась), p-значение по-прежнему остается равным 0. Это говорит о том, что остатки модели по-прежнему не следуют нормальному распределению. Это может указывать на наличие структурных особенностей в данных, которые модель не полностью учитывает.

Также была приняла попытка применить полиномиальную модель степени 2, но её коэффициент детерминации (R^2) составил всего 0.6412, что улучшило предсказательную способность по сравнению с линейной регрессией на логарифмированных данных, но разница незначительна (менее 1%). Это может свидетельствовать о том, что линейная зависимость в логарифмированном пространстве уже хорошо объясняет данные.

3.4 Построение эмпирического моста

Попробуем построить эмпирический мост для прологарифмированных данных:

```
# Остатки от модели
residuals = y - y_pred

# Накопленная сумма остатков
sum_residuals = np.cumsum(residuals)

# Нормировка
empirical_bridge = sum_residuals / len(residuals) ** 0.5 / np.std(residuals)

J = np.max(np.abs(empirical_bridge))
p_value = stats.kstwobign.sf(J)
```

Выходные данные:

```
J_n = 0.5762153709373556
p-value = 0.894123596960542
```

J_n = 0.5762: эта статистика измеряет наибольшее отклонение между эмпирической функцией распределения остатков и теоретической функцией распределения (в данном случае нормальной). Чем выше это значение, тем больше отклонение наблюдается от предполагаемого нормального распределения. Текущее значение может указывать на то, что остатки теперь лучше согласуются с нормальным распределением

p-value = 0.8941: высокое p-значение (более 0.05) указывает на то, что отклонения между эмпирическим и теоретическим распределениями можно считать незначительными, и остатки теперь более близки к нормальному распределению.

Результаты представлены на рисунке 7:

Рисунок 7: Эмпирический мост

Хотя p-значение для теста Шапиро-Уилка маленькое (что указывает на отклонение от нормальности), p-значение для эмпирического моста большое (что говорит о том, что остатки, возможно, не так сильно отклоняются от нормального распределения). Это означает, что остатки все-таки могут быть достаточно близки к нормальному распределению по эмпирическому мосту, несмотря на то, что тест Шапиро-Уилка показал отклонение.

3.5 Разделение данных

В ходе работы было выдвинуто предположение о том, что разделение данных на 2 части по минимуму эмпирического моста может улучшить предсказательную способность модели.

```
# Находим точку минимума эмпирического моста
min_index = np.argmin(empirical_bridge)
min_value = empirical_bridge[min_index]
min_log_weight = log_weights[min_index]

# Разделяем данные на две части по точке минимума
data_before_min = df[df['log_weight_kg'] <= min_log_weight]
data_after_min = df[df['log_weight_kg'] > min_log_weight]
```

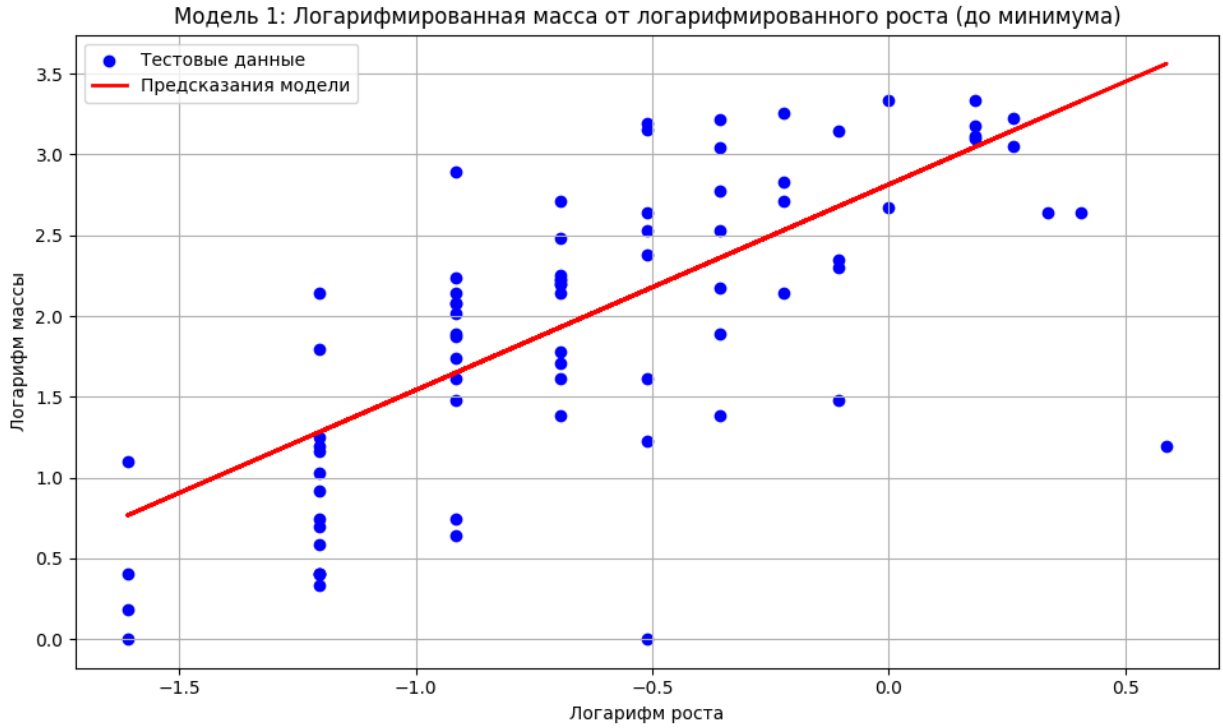

Рисунок 8: Моделирование до минимума эмпирического моста

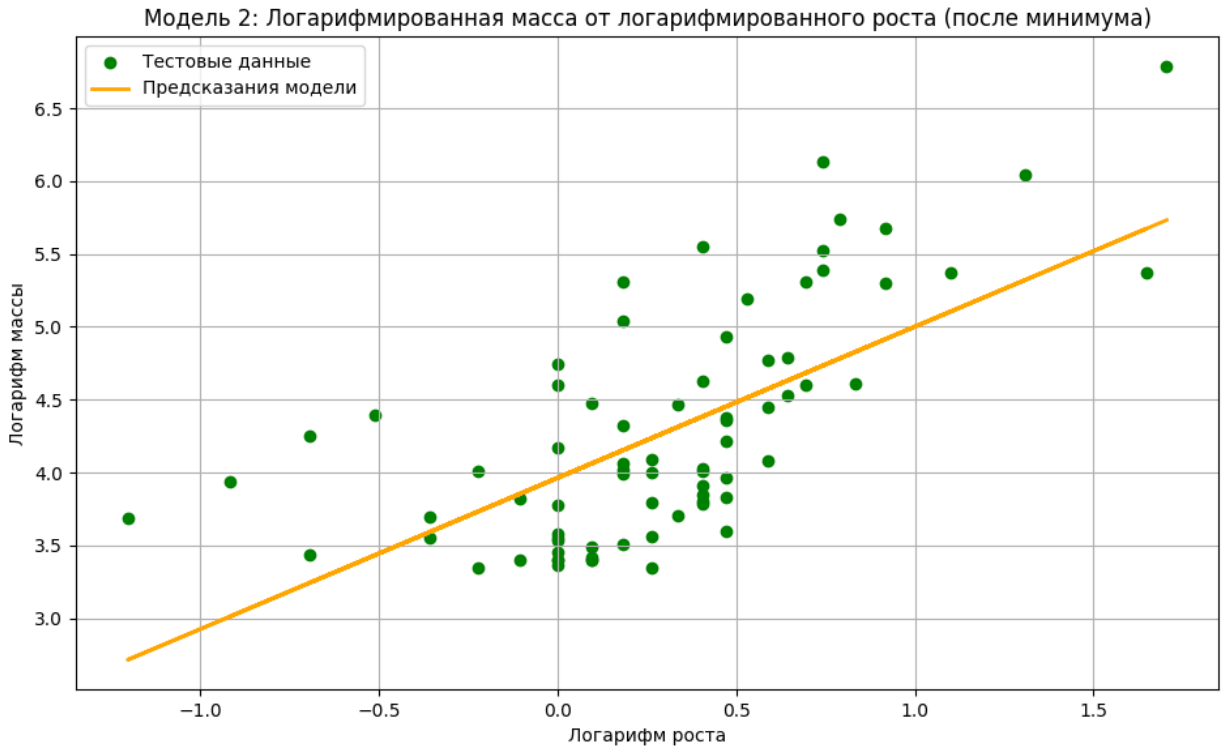

Рисунок 9: Моделирование после минимума эмпирического моста

Выходные данные представлены в таблице 3:

Данные	$\mathbf{R^2}$	**MSE**
Модель до точки минимума	0.4840	0.4373
Модель после точки минимума	0.4482	0.3412

Таблица 3: Результаты аналитического изучения прологарифмированной модели, разделённой по точке минимума эмпирического моста

Модели до и после минимума показывают схожие результаты. Тем не менее, модель после минимума имеет немного более высокий R^2 (0.4482 против 0.4840), что свидетельствует о лучшем объяснении вариации массы покемонов после точки минимума. Однако R^2 для обеих частей выборки хуже, чем R^2 = 0.6399 на тестовых данных, рассчитанный ранее.

4 ВЫВОД

В ходе работы была разработана и оценена вероятностная модель на основе набора исходных данных, применены изученные методы статистического анализа и моделирования для обеспечения максимально возможной для данного набора данных точности модели и её предсказательной способности.

5 ПРИЛОЖЕНИЕ

5.1 Источник данных

Данные для исследования взяты с
https://www.kaggle.com/datasets/rounakbanik/pokemon/data

5.2 Листинг программы на Python

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import shapiro
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv('pokemon.csv')

# Очистка от пропусков
df = df.dropna(subset=['height_m', 'weight_kg'])

# Сортировка данных по весу (height_m) в порядке возрастания
df = df.sort_values(by='height_m').reset_index(drop=True)

# Логарифмическое преобразование признаков
df['log_height_m'] = np.log(df['height_m'])
df['log_weight_kg'] = np.log(df['weight_kg'])

# Очистка от выбросов с использованием IQR
def remove_outliers(df, column_name):
    Q1 = df[column_name].quantile(0.25)
    Q3 = df[column_name].quantile(0.75)
    IQR = Q3 - Q1
```

```
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column_name] >= lower_bound) & (df[column_name] <= upper_bound)]

df = remove_outliers(df, 'log_height_m')
df = remove_outliers(df, 'log_weight_kg')

# Выбор признаков и целевой переменной
X = df[['log_height_m']]
y = df['log_weight_kg']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print(f'Коэффициент модели: {model.coef_[0]}')
print(f'Свободный член (intercept): {model.intercept_}')
print(f'Коэффициент детерминации (R^2) на тестовых данных: {model.score(X_test, y_test)}')

plt.figure(figsize=(10, 6))
sns.scatterplot(x=X_test['log_height_m'], y=y_test, label='Истинные значения')
sns.lineplot(x=X_test['log_height_m'], y=y_pred, color='red', label='Предсказания')
plt.xlabel('Рост (м), логарифм')
plt.ylabel('Масса (кг), логарифм')
plt.title('Зависимость массы покемона от его роста, логарифм')
plt.legend()
plt.show()

# Кросс-валидация
from sklearn.model_selection import KFold
kf = KFold(n_splits=5, shuffle=True, random_state=42)
cross_val_scores = cross_val_score(model, X, y, cv=kf, scoring='r2')

print(f'Средний коэффициент детерминации (R^2)'
      'по результатам кросс-валидации: {np.mean(cross_val_scores):.4f}')
print(f'Стандартное отклонение R^2'
      'по результатам кросс-валидации: {np.std(cross_val_scores):.4f}')
```

```
# Обучение модели на всем наборе данных для анализа остатков
model.fit(X, y)
y_pred = model.predict(X)
residuals = y - y_pred

# Тест Шапиро-Уилка для проверки нормальности распределения остатков
stat, p_value = shapiro(residuals)
print(f'Статистика теста Шапиро-Уилка: {stat:.4f}, p-значение: {p_value:.4f}')

plt.figure(figsize=(10, 6))
sns.histplot(residuals, kde=True, bins=20)
plt.title('Распределение остатков')
plt.xlabel('Остатки (Фактические - Предсказанные значения)')
plt.ylabel('Частота')
plt.show()

# Построение графика остатков
plt.figure(figsize=(10, 6))
plt.scatter(X, residuals, alpha=0.7)
plt.axhline(y=0, color='r', linestyle='--')
plt.xlabel('Рост (м), логарифм')
plt.ylabel('Остатки, логарифм')
plt.title('График остатков')
plt.show()

log_weights = df['log_weight_kg'].values
log_weights = np.sort(log_weights)

# Остатки от модели
residuals = y - y_pred

# Накопленная сумма остатков
sum_residuals = np.cumsum(residuals)

# Нормировка
empirical_bridge = sum_residuals / len(residuals) ** 0.5 / np.std(residuals)

# Статистика и p-value
J = np.max(np.abs(empirical_bridge))
```

```
p_value = stats.kstwobign.sf(J)

print("J_n =", J)
print("p-value =", p_value)

# Визуализация эмпирического моста
plt.figure(figsize=(10, 6))
plt.plot(log_weights, empirical_bridge, label="Эмпирический мост", color='blue')
plt.axhline(0, color='red', linestyle='--', label="Линия идеального распределения")
plt.title("Эмпирический мост для логарифма массы покемонов")
plt.xlabel("Логарифм массы")
plt.ylabel("Отклонение (эмпирический мост)")
plt.legend()
plt.grid()
plt.show()

# Находим точку минимума эмпирического моста
min_index = np.argmin(empirical_bridge)
min_value = empirical_bridge[min_index]
min_log_weight = log_weights[min_index]

print(f"Точка минимума эмпирического моста: {min_log_weight}, отклонение: {min_value}")

# Разделяем данные на две части по точке минимума
# Первая часть данных до точки минимума
data_before_min = df[df['log_weight_kg'] <= min_log_weight]
# Вторая часть - после
data_after_min = df[df['log_weight_kg'] > min_log_weight]

# Первая модель
X_before = data_before_min[['log_height_m']]
y_before = data_before_min['log_weight_kg']
X_train_before, X_test_before, y_train_before, y_test_before = train_test_split(X_before, y_before
model_before = LinearRegression()
model_before.fit(X_train_before, y_train_before)
y_pred_before = model_before.predict(X_test_before)

# Вторая модель
X_after = data_after_min[['log_height_m']]
y_after = data_after_min['log_weight_kg']
```

```
X_train_after, X_test_after, y_train_after, y_test_after = train_test_split(X_after, y_after, test
model_after = LinearRegression()
model_after.fit(X_train_after, y_train_after)
y_pred_after = model_after.predict(X_test_after)

# Оценка моделей
mse_before = mean_squared_error(y_test_before, y_pred_before)
r2_before = r2_score(y_test_before, y_pred_before)
mse_after = mean_squared_error(y_test_after, y_pred_after)
r2_after = r2_score(y_test_after, y_pred_after)

print(f"Модель 1 (до минимума): R^2 = {r2_before}, MSE = {mse_before}")
print(f"Модель 2 (после минимума): R^2 = {r2_after}, MSE = {mse_after}")

plt.figure(figsize=(12, 6))
plt.scatter(X_test_before, y_test_before, color='blue', label="Тестовые данные")
plt.plot(X_test_before, y_pred_before, color='red', linewidth=2, label="Предсказания модели")
plt.title("Модель 1: Логарифмированная масса от логарифмированного роста (до минимума)")
plt.xlabel("Логарифм роста")
plt.ylabel("Логарифм массы")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()

plt.figure(figsize=(12, 6))
plt.scatter(X_test_after, y_test_after, color='green', label="Тестовые данные")
plt.plot(X_test_after, y_pred_after, color='orange', linewidth=2, label="Предсказания модели")
plt.title("Модель 2: Логарифмированная масса от логарифмированного роста (после минимума)")
plt.xlabel("Логарифм роста")
plt.ylabel("Логарифм массы")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```