МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГАОУ ВО «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ ИНФОРМАЦИОННЫХ

ТЕХНОЛОГИЙ И ТЕЛЕКОММУНИКАЦИЙ

КАФЕДРА ПРИКЛАДНОЙ ИНФОРМАТИКИ

**Лабораторная работа №10**

по дисциплине

«Искусственный интеллект и машинное обучение»

**Выполнил:**

Сердюков Никита Анатольевич

Студент 2 курса группы \_ПИН-б-о-22-1

Направления подготовки

09.03.03 Прикладная информатика

очной формы обучения

Ставрополь, 2023 г.

Тема: ПОЛИНОМИАЛЬНАЯ РЕГРЕССИЯ

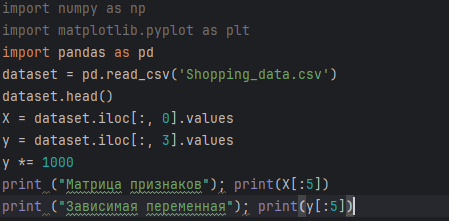
Цель работы: научиться применять разработанный пайплайн для тиражирования кода с целью решения задачи полиномиальной регрессии

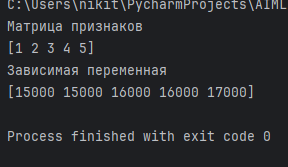
Выполнение работы:

Датасет: Shopping dataset

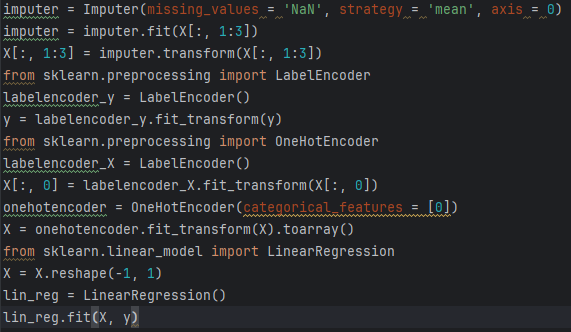
Описание датасета: Идентификатор клиента (Customer ID): уникальный идентификатор, присваиваемый каждому клиенту для целей отслеживания и анализа.

1. Загрузка данных





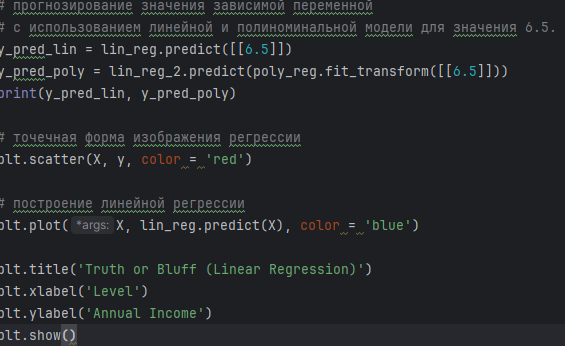
1. Обработка пропущенных значений, преобразования категориальных меток в числовые значения обучение линейной модели

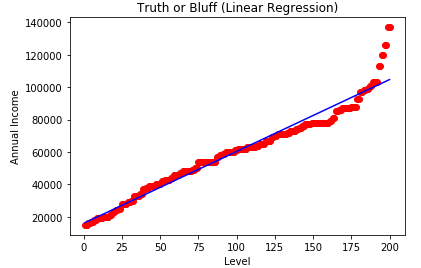


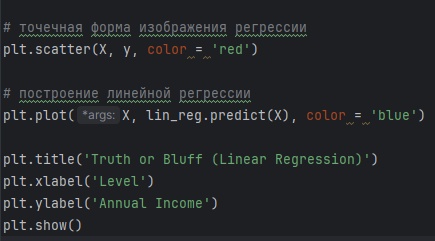
Полиномиальная модель - это модель, которая используется для замены (аппроксимации) сложных нелинейных зависимостей между переменными. В отличие от линейной модели, которая предполагает линейную зависимость между предикторами и зависимой переменной, полиномиальная модель позволяет учитывать криволинейные отношения путем включения степеней их значений в модель.

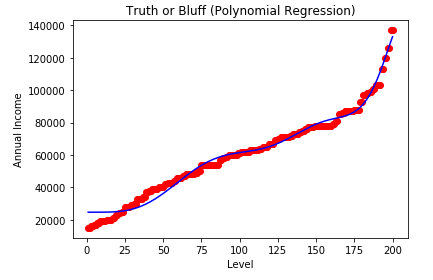
Например, в полиномиальной модели вместо простой линейной зависимости типа y = ax + b могут использоваться более сложные уравнения вида y = ax^2 + bx + c или даже более высоких степеней.

1. Предсказание, обработка и визуализация результатов

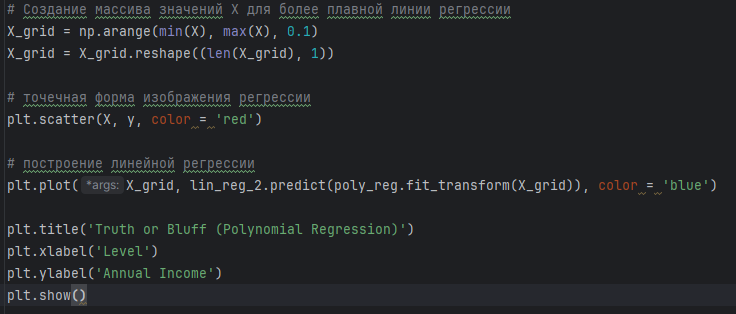


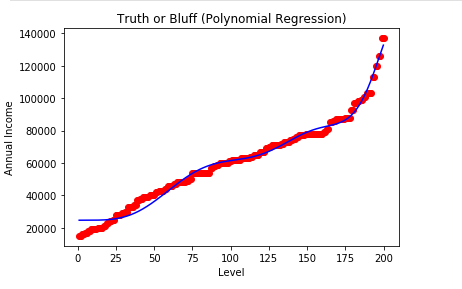


1. 



1. Создание массива значений X для более плавной линии регрессии





Листинг:

import numpy as np  
import matplotlib.pyplot as plt  
import pandas as pd  
dataset = pd.read\_csv('Shopping\_data.csv')  
dataset.head()  
X = dataset.iloc[:, 0].values  
y = dataset.iloc[:, 3].values  
y \*= 1000  
print ("Матрица признаков"); print(X[:5])  
print ("Зависимая переменная"); print(y[:5])  
from sklearn.preprocessing import Imputer  
imputer = Imputer(missing\_values = 'NaN', strategy = 'mean', axis = 0)  
imputer = imputer.fit(X[:, 1:3])  
X[:, 1:3] = imputer.transform(X[:, 1:3])  
from sklearn.preprocessing import LabelEncoder  
labelencoder\_y = LabelEncoder()  
y = labelencoder\_y.fit\_transform(y)  
from sklearn.preprocessing import OneHotEncoder  
labelencoder\_X = LabelEncoder()  
X[:, 0] = labelencoder\_X.fit\_transform(X[:, 0])  
onehotencoder = OneHotEncoder(categorical\_features = [0])  
X = onehotencoder.fit\_transform(X).toarray()  
from sklearn.linear\_model import LinearRegression  
X = X.reshape(-1, 1)  
lin\_reg = LinearRegression()  
lin\_reg.fit(X, y)  
from sklearn.preprocessing import PolynomialFeatures  
poly\_reg = PolynomialFeatures(degree = 10)  
X\_poly = poly\_reg.fit\_transform(X)  
poly\_reg.fit(X\_poly, y)  
lin\_reg\_2 = LinearRegression()  
lin\_reg\_2.fit(X\_poly, y)  
# прогнозирование значения зависимой переменной  
# с использованием линейной и полиноминальной модели для значения 6.5.  
y\_pred\_lin = lin\_reg.predict([[6.5]])  
y\_pred\_poly = lin\_reg\_2.predict(poly\_reg.fit\_transform([[6.5]]))  
print(y\_pred\_lin, y\_pred\_poly)  
  
# точечная форма изображения регрессии  
plt.scatter(X, y, color = 'red')  
  
# построение линейной регрессии  
plt.plot(X, lin\_reg.predict(X), color = 'blue')  
  
plt.title('Truth or Bluff (Linear Regression)')  
plt.xlabel('Level')  
plt.ylabel('Annual Income')  
plt.show()  
# Создание массива значений X для более плавной линии регрессии  
X\_grid = np.arange(min(X), max(X), 0.1)  
X\_grid = X\_grid.reshape((len(X\_grid), 1))  
  
# точечная форма изображения регрессии  
plt.scatter(X, y, color = 'red')  
  
# построение линейной регрессии  
plt.plot(X\_grid, lin\_reg\_2.predict(poly\_reg.fit\_transform(X\_grid)), color = 'blue')  
  
plt.title('Truth or Bluff (Polynomial Regression)')  
plt.xlabel('Level')  
plt.ylabel('Annual Income')  
plt.show()

Вывод: научился применять разработанный пайплайн для тиражирования кода с целью решения задачи полиномиальной регрессии.