قراره یه برنامه با پایتون بنویسم که این کارها رو بکنه:

۱. از یه وبسایت املاک تعداد زیادی از مشخصات برای هر ملک رو استخراج کنم (بیش از ۱۰ هزار ملک)

۲. داده های بدست اومده رو پاکسازی کنیم ۳. داده ها رو انالیز کنیم ۴. نتیجه انالیز ها رو با نمودار نشون بدیم

۵. شاخص قیمت هر متر ملک بر اساس تقسیم بندی منطقی در بلزیک‌رو از سایت اطلاع رسانی دولتی استخراج کنه

۶. قیمت رو بر اساس شاخ و بر اساس اطلاعات استخراج شده محاسبه کنه

۷. یه مدل یادگیری هوش مصنوعی ایجاد کنه که شیوه قیمت گداری رو یاد بگیره و بر اساس فاکتورهای موجود قیمت رو دقیقتر محاسبه کنه

۸. قیمتهای ملک در بازه های زمانی محتلف رو بگیره و قیمتها رو با مدل یادگیری ماشین پیشبینی کنه

۹. درنهایت یه رابط کاربری قابل اجرا برای استفاده در وبسات براش ساخته بشه

. همه این کارها باید انجام بشه . تمام قسمتها بر اساس شماره گذاری یکی یکی اضافه میشن به برنامه

. چندتا سوال:

۱. بهترین مدل برای نوشتن این برنامه چیه ۲. از همین الان چه توابع یا کلاس با هرچیزی رو که لازمه باید بسازیم؟

۳ فونداسیون برنامه رو چطور پی‌ریزی کنم که بعدا قابل توسعه باشه و بهتر بشه ازش استفاده کرد و عملکرد بهتری داشته باشه؟

برای انجام چنین پروژه‌ای که چندین مرحله مختلف شامل استخراج داده، پاکسازی، تحلیل، یادگیری ماشین، و ساخت رابط کاربری را در بر می‌گیرد، باید برنامه‌ریزی دقیقی داشته باشید.

**۱. بهترین مدل برای نوشتن این برنامه چیست؟**

برای چنین پروژه‌ی چند بخشی، بهترین مدل توسعه، استفاده از معماری **ماژولار** و **قابل گسترش** است. به این معنی که هر بخش از برنامه را به صورت جداگانه پیاده‌سازی کنید و آنها را از طریق رابط‌های مشخص به هم متصل کنید. این کار به شما کمک می‌کند که هر بخش را به طور مستقل توسعه داده، تست کرده و نگهداری کنید.

برنامه باید شامل چندین ماژول باشد:

* **ماژول وب اسکرپینگ (Web Scraping)**: برای استخراج داده از وبسایت‌های املاک.
* **ماژول پاکسازی داده (Data Cleaning)**: برای پردازش و پاکسازی داده‌ها.
* **ماژول تحلیل داده (Data Analysis)**: برای آنالیز داده و نمایش نمودارها.
* **ماژول استخراج داده‌های خارجی**: برای دریافت شاخص‌های قیمت از وب‌سایت‌های دولتی.
* **ماژول یادگیری ماشین (Machine Learning)**: برای ساخت مدل‌های پیش‌بینی قیمت.
* **ماژول رابط کاربری (Frontend/Backend)**: برای ارائه رابط کاربری وب.

بهترین روش این است که از **طراحی مبتنی بر API** استفاده کنید تا بتوانید بخش‌های مختلف را به سادگی به هم متصل کرده و در صورت لزوم آنها را به‌روزرسانی کنید.

**۲. از همین الان چه توابع یا کلاس‌هایی را باید بسازیم؟**

برای پی‌ریزی یک فونداسیون قوی و قابل توسعه، باید چندین کلاس و توابع پایه را از ابتدا تعریف کنید. در ادامه لیستی از ماژول‌ها و کلاس‌های پیشنهادی را می‌آورم:

**الف) ماژول وب اسکرپینگ** برای استخراج داده از وبسایت‌ها از کتابخانه‌هایی مانند BeautifulSoup، Scrapy یا Selenium استفاده کنید.

class PropertyScraper:

def \_\_init\_\_(self, url):

self.url = url

def get\_page(self):

"""Fetches the webpage from the given URL."""

# Code to fetch the webpage

def parse\_properties(self, page\_content):

"""Parses the HTML and extracts property data."""

# Code to parse the page and extract data

def save\_data(self, data):

"""Saves the extracted data into a CSV or database."""

# Code to save data

**ب) ماژول پاکسازی داده** این بخش داده‌های استخراج شده را بررسی و داده‌های ناکارآمد یا نادرست را حذف می‌کند.

class DataCleaner:

def \_\_init\_\_(self, data):

self.data = data

def remove\_nulls(self):

"""Removes rows with null or invalid values."""

# Code to clean data

def normalize\_data(self):

"""Normalizes and standardizes data (e.g., convert units)."""

# Code to normalize data

**ج) ماژول تحلیل داده** برای آنالیز و رسم نمودارها، از کتابخانه‌هایی مانند Pandas برای تحلیل داده و Matplotlib یا Plotly برای نمایش نمودارها استفاده کنید.

class DataAnalyzer:

def \_\_init\_\_(self, cleaned\_data):

self.cleaned\_data = cleaned\_data

def analyze(self):

"""Performs data analysis and returns insights."""

# Code to analyze the data

def plot(self):

"""Generates charts and plots based on the analysis."""

# Code to generate charts

**د) ماژول شاخص قیمت** این ماژول داده‌های شاخص قیمت را از وبسایت‌های دولتی دریافت می‌کند------------------------------------------

class PriceIndexFetcher:

def \_\_init\_\_(self, government\_url):

self.government\_url = government\_url

def fetch\_index(self):

"""Fetches the latest price index for different regions."""

# Code to fetch data

**هـ) ماژول یادگیری ماشین** برای پیاده‌سازی مدل یادگیری ماشین، از کتابخانه‌هایی مانند Scikit-learn یا TensorFlow استفاده کنید-----------

class PricePredictionModel:

def \_\_init\_\_(self, data):

self.data = data

self.model = None

def train\_model(self):

"""Trains a machine learning model on the given data."""

# Code to train the model

def predict\_price(self, new\_data):

"""Predicts property prices based on new input data."""

# Code to predict price

**و) ماژول رابط کاربری** برای ساخت رابط کاربری، می‌توانید از فریمورک‌هایی مانند **Flask** یا **Django** استفاده کنید ---------------------

from flask import Flask, render\_template

app = Flask(\_\_name\_\_)

@app.route('/')

def home():

"""Renders the home page."""

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

"""Handles property price predictions."""

# Code to get input data and return price predictions

**۳. فونداسیون برنامه را چطور پی‌ریزی کنیم؟**

برای پی‌ریزی فونداسیون برنامه باید به موارد زیر توجه کنید:

1. **طراحی ماژولار**: هر بخش از برنامه باید به صورت مستقل باشد و تنها از طریق API یا کلاس‌های تعریف شده با سایر بخش‌ها ارتباط برقرار کند. این کار توسعه، تست و نگهداری را بسیار ساده‌تر می‌کند.
2. **مدیریت استثناها و خطاها**: از ابتدا به مدیریت خطاها در هر ماژول توجه کنید. مثلاً در ماژول اسکرپینگ وب باید استثناهایی نظیر عدم دسترسی به وبسایت یا خطاهای شبکه را مدیریت کنید.
3. **ذخیره‌سازی داده**: داده‌ها باید به صورت مناسب در دیتابیس ذخیره شوند. از دیتابیس‌های SQL مانند **PostgreSQL** یا دیتابیس‌های NoSQL مانند **MongoDB** می‌توانید استفاده کنید. بهتر است داده‌ها به گونه‌ای ذخیره شوند که در آینده برای تحلیل‌ها و بهبود مدل‌های یادگیری ماشین به سادگی در دسترس باشند.
4. **مقیاس‌پذیری (Scalability)**: طراحی باید به گونه‌ای باشد که در صورت افزایش حجم داده‌ها یا نیاز به پردازش‌های سنگین‌تر، به راحتی قابل گسترش باشد. برای این منظور، می‌توانید از تکنیک‌های پردازش موازی یا حتی سرویس‌های مبتنی بر کلود استفاده کنید.
5. **نگهداری و توسعه آسان**: نوشتن تست‌های واحد برای هر بخش از برنامه از ابتدا کمک زیادی به پایداری و توسعه خواهد کرد. از ابزارهایی مانند **pytest** برای نوشتن تست‌های واحد استفاده کنید.

با این فونداسیون، هر بخش از برنامه قابل گسترش و بهبود است. می‌توانید ابتدا با ماژول‌های ابتدایی مثل اسکرپینگ و پاکسازی داده شروع کنید و سپس به تدریج ماژول‌های یادگیری ماشین و رابط کاربری را اضافه کنید.