پروژه نهایی درس داده‌کاوی

عنوان پروژه: «علموص‌یار» – داشبورد تحلیلی و سیستم پیشنهاددهنده اساتید

---

۱. ایده‌ی کلی پروژه «علموص‌یار» 🤝

داستان از کجا شروع می‌شه؟
همه‌مون موقع انتخاب واحد می‌ریم سراغ کانال «اساتید علموص» تو تلگرام تا ببینیم بچه‌ها درباره‌ی هر استاد چی گفتن 😅
خودِ همین کانال در واقع یه دیتاست خفن و واقعی‌ـه، فقط خیلی شلخته و پراکنده‌ست.

کاری که تو این پروژه باید بکنید اینه که:

- پیام‌های خام کانال (فایل JSON) رو
  تبدیل کنید به **یه داشبورد تحلیلی + یه سیستم پیشنهاددهنده‌ی استاد/درس** 🎯

هدف نهایی:

- به دانشجو کمک کنه راحت‌تر استاد/درس مناسب خودش رو پیدا کنه ✅

- به اساتید و مدیرگروه‌ها یه دید کلی از بازخورد دانشجوها بده ✅

تمرکز پروژه روی این سه تا محوره:

1. **مهندسی داده + NLP**
   Parsing، Cleaning، Normalization، Sentiment، Keywords، Clustering و ...
2. **مدل‌های داده‌کاوی**
   نظارتی / غیرنظارتی، خوشه‌بندی، insightها و رابطه بین متغیرها
3. **محصول نهایی**
   یه داشبورد تعاملی که یه دانشجوی عادی هم بفهمه چی به چیه + سیستم پیشنهاددهنده

---

۲. خروجی نهایی قراره چی باشه؟ 🚀

در نهایت به **یک وب‌اپلیکیشن تحلیلی (داشبورد) + سیستم پیشنهاددهنده‌ی استاد/درس** برسیم که روی داده‌های واقعی کانال کار کنه.

کاربری که میاد تو سیستم (دانشجو یا مدیر آموزشی) باید بتونه:

🔍 جستجو (Search)

- بر اساس **نام استاد، فامیل یا اسم درس** سرچ کنه.

🎛️ فیلتر (Filter)

- فیلتر کنه بر اساس:
  - دانشکده
  - درس
  - ترم
  - میانگین امتیازها
  - وضعیت نمره‌دهی
  - و ...

👩‌🏫 پروفایل استاد (Professor Profile)

برای هر استاد یه نمای کلی (۳۶۰ درجه) ببینه، شامل:

- میانگین ۶ امتیاز عددی (تدریس، مدیریت کلاس، تسلط، تعامل، پاسخ‌گویی و...)
- تحلیل وضعیت نمره‌دهی (مثلاً: ۷۰٪ منصفانه، ۲۰٪ سخت‌گیر، ۱۰٪ آسان)
- تحلیل متن کامنت‌ها:
  - امتیاز احساسات (Sentiment)
  - ابر کلمات (Word Cloud)
- روند امتیاز و رضایت در ترم‌های مختلف (اگه اطلاعات زمانی داشته باشین)

⚖️ مقایسه‌ی اساتید (Compare)

- چندتا استاد/درس رو انتخاب کنه و کنار هم مقایسه‌شون کنه:
  - میانگین امتیازها
  - وضعیت نمره‌دهی
  - sentiment و ...

🤖 سیستم پیشنهاددهنده (Recommender)

بر اساس ترجیحات دانشجو، مثلاً:

- استاد/درس‌هایی که دوست داره
- اولویت‌ها:
  - نمره‌دهی آسان
  - امتحان سبک
  - پروژه‌محور بودن
  - تدریس قوی
  - و ...

یه لیست **اساتید/درس‌های پیشنهادی** برگردونید.

مدل پیشنهاددهنده می‌تونه مثلاً:

- Content-Based باشه
- Collaborative Filtering ساده باشه
- یا حتی Rule-based ساده

مهم اینه که:

- منطقی باشه ✅
- بشه توضیح داد چرا این استادها پیشنهاد شدن ✅
- تو گزارش مستند شده باشه ✅

---

۳. داده از کجا میاد؟ 📁

۳.۱. منبع اصلی

بهتون **یه فایل JSON** می‌دیم که خروجی کامل پیام‌های کانال علموص اساتیده.
این می‌شه دیتای خام پروژه‌تون. (دانلود فایل JSON)

۳.۲. ساختار کلی

هر رکورد JSON = یک پیام کانال. توش اینا رو داریم:

- **فراداده (Metadata):**

    - شناسه پیام، تاریخ/ترم (اگه هست)، اطلاعات کانال
- **متن پیام (message_text):**
    - اسم استاد
    - دانشکده
    - اسم درس
    - ۶ تا امتیاز عددی (از ۱۰)
    - توضیحات درباره‌ی نمره‌دهی، حضور و غیاب و... (متن آزاد)
- **کامنت/توضیح دانشجو** (متن بدون ساختار)

۳.۳. امتیاز اضافه ✨

اگه خواستین بهترش کنین:

- به‌جای استفاده از JSON آماده، خودتون با **API تلگرام** (Telethon / Pyrogram) داده رو بکشید بیرون. این‌جوری داده‌های داشبورد همیشه به‌روز می‌مونن.
- اگه این کار رو می‌کنید، تو گزارش باید توضیح بدین:
    - چطوری وصل شدین
    - چطوری داده جمع کردین
    - چطور ذخیره و مدیریت کردین

---

۴. فازهای پروژه؛ قدم‌به‌قدم 🛠️

این یه roadmap پیشنهادی‌ه؛ لازم نیست مو به مو همین باشه، ولی در این حد کار باید انجام بشه.

فاز ۱: بارگذاری، اکتشاف و Parsing (Load, EDA & Parsing) 🔎

✅ *تسک ۱ – بارگذاری و EDA*

- خوندن JSON و تبدیلش به یه DataFrame تو Pandas

- فهمیدن ساختار داده:
    - چند تا پیام داریم؟
    - حدوداً چندتا استاد مختلف داریم؟
    - توزیع امتیازها چطوریه؟
    - چقدر داده‌ی گمشده داریم؟
- کشیدن چند نمودار اکتشافی:

  - هیستوگرام امتیازها
  - تعداد نظرات به تفکیک دانشکده و ...
- مستندسازی EDA تو یه نوت‌بوک جدا (با توضیحات متنی)

✅ *تسک ۲ – نوشتن Parser*

یه تابع/کلاس پایتون که:

- متن خام پیام رو بگیره
- در عوض یه دیکشنری مرتب بده، شامل حداقل:
  - professor_id (بعداً ساخته می‌شه)
  - professor_name_raw
  - department
  - course_name
  - rating_1 تا rating_6
  - grading_status_raw
  - attendance_status_raw
  - comment_text
  - term یا date (اگه قابل استخراج باشه)

نکات مهم:

- احتمالا لازمه از **Regex** و الگوهای متنی استفاده کنید.
- پارسر باید نسبت به تغییرات ریز (ایموجی، جابه‌جایی خط‌ها، فاصله‌ها و...) مقاوم باشه.
- اگه جایی نتونست درست parse کنه، اون پیام رو تو یه ستون مثل parse_error لاگ کنه.
- برای پارسر چندتا **Unit Test** بنویسید (روی نمونه‌ها و edge caseها).

فاز ۲: پاک‌سازی و استانداردسازی (Data Cleaning & Standardization) 🧹

✅ *تسک ۳ – یکی‌کردن اسامی اساتید*

چالش:

- «دکتر الف. رضایی»
- «علیرضا رضایی»
- «رضایی»

- «A. Rezaei»

اینا شاید همه یک نفر باشن 😅

راه‌حل پیشنهادی:

- نرمال‌سازی اولیه از جمله:
  - حذف فاصله‌های اضافی
  - یکسان‌سازی حروف عربی/فارسی
  - lowercase برای اسامی لاتین
- ترکیب **دانشکده + الگوریتم‌های شباهت رشته‌ای** (FuzzyWuzzy، Levenshtein و...)
- در نهایت گروه‌بندی اسامی مشابه و ساختن یک professor_id یکتا برای هر استاد.

✅ *تسک ۴ – یکی‌کردن اسم درس و برچسب‌های متنی*

- یکی‌کردن اسم درس‌ها:
  - «مبانی طراحی ۲» و «مبانی طراحی 2» ← یه درس واحد
- استانداردسازی وضعیت نمره‌دهی و حضور و غیاب:
  - یه سری برچسب تعریف کنید مثل:
    - نمره‌دهی: {آسان، منصفانه، سخت‌گیر، نامشخص}
    - حضور و غیاب: {سخت‌گیر، متوسط، آزاد، نامشخص}
  - متن‌های آزاد مثل «نمره نمیده»، «نمره‌اش خوبه» رو مپ کنید به این برچسب‌ها.

فاز ۳: NLP، Feature Engineering و مدل‌ها 🤖

اینجا پروژه از حالت «گزارش ساده» می‌ره تو مود **تحلیل داده‌کاوانه جدی**.

۴.۳.۱. تحلیل احساسات (Sentiment Analysis) 💔❤️

ورودی: ستون comment_text

حداقل کار:

1. پیش‌پردازش متن فارسی (Shekar یا موارد مشابه):
   - نرمال‌سازی
   - توکن‌سازی
   - حذف stopwordها
2. استفاده از یک روش ساده برای Sentiment:

- روش‌های Ruled-Based (به شرطی که عملکرد و دقت قابل تحملی روی داده واقعی داشته باشه)

- یا یه مدل آماده‌ی سبک (مثلاً یه مدل HuggingFace که روی CPU جواب بده)

خروجی:
یه ستون جدید مثل sentiment_score (مثلاً [-1,1]) یا برچسب {منفی، خنثی، مثبت}.

✨ امتیاز اضافه: استفاده از مدل‌های Transformer فارسی مثل ParsBERT و گزارش نتایج.

۴.۳.۲. کلمات کلیدی و Topicها 🗣️

برای هر استاد:

- متن‌ها رو نرمال‌سازی کنید + stopword حذف کنید.
- با Frequency یا TF-IDF کلمات کلیدی رو دربیارید.
- ازش استفاده کنید برای:
  - ساخت ابر کلمات
  - شناخت کلی حال‌وهوای کامنت‌ها

✨ امتیاز اضافه: استفاده از Topic Modeling مثل LDA یا BERTopic برای پیدا کردن موضوعات اصلی نظرات.

۴.۳.۳. یک مدل نظارتی (Supervised) – اجباری ✅

باید حداقل یه مسئله‌ی Supervised تعریف و حل کنید. مثلاً:

- پیش‌بینی **overall_score** از روی:
  - میانگین ۶ امتیاز
  - وضعیت نمره‌دهی (کدگذاری‌شده)
  - میانگین sentiment

  یا:
- طبقه‌بندی نظرات به {مثبت/خنثی/منفی} از روی ویژگی‌های عددی و متنی.

حداقل‌ها:

- Train/Test (یا یه Cross-Validation ساده)
- حداقل یک مدل ساده مثل:
  - Logistic Regression
  - Random Forest
  - SVM ساده
- گزارش معیارهای ارزیابی مناسب (Accuracy، F1، MAE، RMSE و...)
- یه تحلیل کوچیک از Feature Importance یا حداقل تفسیر منطقی نتایج.

۴.۳.۴. یک مدل غیرنظارتی (Unsupervised) – اجباری ✅

مثال خوب: **خوشه‌بندی اساتید** بر اساس پروفایلشون:

- ورودی:
  میانگین ۶ امتیاز + وضعیت نمره‌دهی + میانگین sentiment + چند feature ساده دیگه
- روش‌ها:
  - KMeans
  - Agglomerative Clustering
- خروجی و نمایش:
  - بعد رو کم کنید (مثلاً PCA)
  - هر استاد رو به‌صورت یه نقطه روی نمودار دوبعدی نشون بدین
  - برای هر خوشه یه تفسیر انسانی بدین مثل:
    - «سخت‌گیر ولی محبوب»
    - «آسان‌گیر با رضایت متوسط»
    - و...

۴.۳.۵. سیستم پیشنهاددهنده‌ی استاد/درس – اجباری ✅

باید یه سیستم پیشنهاددهنده‌ی **کارا و منطقی** بسازید. می‌تونه ساده باشه.

چند سناریو:

- **Content-Based Filtering:**
  - برای هر استاد یه بردار ویژگی بسازید (امتیازها، sentiment، کلمات کلیدی)

- با شباهت (مثلاً cosine similarity) اساتید مشابه یه استاد/درس مورد علاقه رو پیشنهاد بدین

- **Rule-Based:**
  - بر اساس فیلترهای انتخابی کاربر (امتیاز بالای ۸، نمره‌دهی منصفانه، پروژه‌محور بودن و...) اساتید رو رتبه‌بندی و پیشنهاد بدین
- **Hybrid ساده:**
  - ترکیبی از امتیازات عددی، sentiment و شباهت کلمات

مهم اینا‌ست که:

- سیستم منطقی باشه
- تو داشبورد شفاف باشه «چرا» این استادها پیشنهاد شدن
- تو گزارش طراحی و جایگزین‌هاش توضیح داده بشه ✅

فاز ۴: ساخت داشبورد / Web App 📊

ابزار پیشنهادی

- پیشنهاد اصلی: **Streamlit**
  چون پایتون‌محور، سریع، راحت و خوب برای DataFrameه.

صفحات حداقلی داشبورد

1. **صفحه‌ی اصلی (Overview):**
   - چند تا کارت شاخص:
     - تعداد کل نظرات
     - تعداد اساتید پوشش‌داده‌شده
     - میانگین امتیاز کلی
   - یکی دو نمودار:
     - هیستوگرام توزیع امتیازها
     - توزیع وضعیت نمره‌دهی
2. **صفحه‌ی جستجو و فیلتر:**
   - جستجوی استاد/درس
   - فیلتر بر اساس دانشکده، ترم، بازه‌ی امتیاز، وضعیت نمره‌دهی

- لیست اساتید مطابق جستجو، با یه خلاصه‌ی کوچیک از هرکدوم

3. **صفحه‌ی پروفایل استاد:**
    - نمودار راداری برای ۶ امتیاز عددی
    - نمودار دایره‌ای/دونات برای نمره‌دهی و حضور و غیاب
    - Word Cloud مثبت و منفی
    - خلاصه‌ی sentiment در ترم‌های مختلف (اگه داده باشه)
4. **صفحه‌ی مقایسه (Compare):**
    - انتخاب چند استاد
    - نمودار میله‌ای برای مقایسه امتیازها
    - یه جدول خلاصه مقایسه
5. **صفحه‌ی پیشنهاددهنده (Recommender):**
    - کاربر ترجیح‌هاش رو وارد کنه (مثلاً تدریس مهم‌تره یا نمره‌دهی آسان)
    - سیستم یه لیست مرتب از اساتید/درس‌های پیشنهادی بده
    - کنار هر پیشنهاد یه توضیح کوتاه مثل:
        - «میانگین تدریس ۹.۲، نمره‌دهی منصفانه، sentiment بالا» 🌟

---

۵. حداقل‌ها در مقابل کارهای خفن‌تر 😎

۵.۱. حداقل کارهایی که باید انجام بشه

برای نمره‌ی خوب (نه لزوماً فول‌مارک)، حداقل:

- تبدیل JSON به DataFrame تمیز با ستون‌های اصلی
- یکپارچه‌سازی نسبی اسم اساتید، درس‌ها، وضعیت نمره‌دهی
- یه تحلیل Sentiment حداقلی روی نظرات
- حداقل یک مدل Supervised + یک مدل Unsupervised
- یک سیستم پیشنهاددهنده‌ی ساده و منطقی
- داشبورد با صفحات حداقلی که گفتیم
- README واضح و قابل اجرا
- کار با Gitو commitهای مرتب و معنادار

۵.۲. چیزهایی که نمره‌ی طلایی می‌آره ✨

امتیاز اضافه برای تیم‌هایی که:

- از مدل‌های NLP پیشرفته مثل ParsBERT استفاده کنن و مستند کنن
- سیستم پیشنهاددهنده‌ی خلاقانه‌تر/Hybrid واقعی پیاده کنن
- تحلیل‌های تحقیقی جذاب ارائه بدن مثل:
  - رابطه‌ی سخت‌گیری نمره‌دهی و رضایت دانشجو
  - پیدا کردن «اساتید underrated» یا «اساتید پرریسک»
  - خوشه‌بندی اساتید و تفسیر باکیفیت خوشه‌ها
- پروژه رو با Docker یا اسکریپت‌های تمیز بسته‌بندی کنن

---

۶. بخش «۳ تصمیم مهم طراحی» 🧠

تو گزارش نهایی یه بخشی باید باشه به اسم:

«۳ تصمیم مهم طراحی (Design Decisions) که در پروژه گرفتیم»

برای هر تصمیم باید بنویسید:

- چیکار کردین و کجای سیستم بوده (پارسر، مدل، داشبورد، Recommender و...)
- چه گزینه‌های جایگزینی داشتین (مثلاً BERT vs TF-IDF، Streamlit vs Dash، بدون fuzzy matching و...)
- چرا این گزینه رو انتخاب کردین و اون یکی رو رد کردین

استدلال‌هاتون هم باید:

- به نتایج تجربی، محدودیت‌ها (زمان، منابع، پیچیدگی) یا تجربه‌ی خودتون تکیه کنه
- توضیح خیلی ساده مثل «این راحت‌تر بود» کافی نیست 😄

---

۷. ملاحظات اخلاقی و Bias ⚖️

چون داده‌ها مربوط به **آدم‌های واقعی** (اساتید)ـه، حتماً تو گزارش نهایی حداقل ۱–۲ پاراگراف درباره‌ی اینا بنویسید:

- چه Biasهایی ممکنه تو داده باشه؟

    - مثلاً ناراضی‌ها بیشتر نظر می‌دن؟
    - بعضی اساتید کمتر دیده می‌شن تو کانال؟
- خطرات استفاده‌ی نادرست از این سیستم چیه؟
    - برچسب‌زدن ناعادلانه
    - تقویت شایعات
- اگه این محصول واقعاً مستقر بشه، چی‌کار می‌شه کرد که:
    - استفاده ازش منصفانه‌تر و مسئولانه‌تر باشه؟

---

۸. تحویل‌ها و ارزیابی 📦

۸.۱. چی باید تحویل بدین؟

- **پروپوزال اولیه (۱–۲ صفحه):**
    - ایده‌ی تیم
    - ابزارهایی که قراره استفاده کنید و چراییش
    - طرح کلی Data Pipeline و ساختار اولیه داشبورد
- **نوت‌بوک‌های پایتون:**
    - NLP، Cleaning، Parsing، EDA، مدل‌های Supervised/Unsupervised
    - با توضیحات متنی (Markdown) کافی
- **کد نهایی اپ (App):**
    - سورس داشبورد + سیستم پیشنهاددهنده (Streamlit/Dash/...)
    - ساختار تمیز (پوشه‌های app ,src/ ,data/ و ...)
- **README.md:**
    - قدم‌به‌قدم راه‌اندازی محیط (virtualenv/conda، نصب پکیج‌ها)
    - چطور نوت‌بوک‌ها و اپ رو اجرا کنیم
    - پیش‌نیازها (نسخه پایتون و ...)
- **گزارش نهایی (Report):**
    - خلاصه EDA و چالش‌های داده
    - توضیح پارسر و Cleaning
    - مدل‌های داده‌کاوی (Supervised / Unsupervised + Recommender)
    - بخش «۳ تصمیم مهم طراحی»
    - بخش Bias و ملاحظات اخلاقی

- **ریپو Git (ترجیحاً GitHub):**
  - لینک ریپو + README + تاریخچه‌ی منطقی commitها

۸.۲. نمره‌دهی تقریبی 📊

- ۳۵٪ – محصول نهایی:
  - آیا داشبورد و Recommender واقعاً کار می‌کنن؟
  - UX/UI قابل قبوله؟
  - برای یه دانشجوی معمولی مفیده؟
- ۳۰٪ – داده، NLP و مدل‌ها:
  - کیفیت پارسر و Cleaning
  - هوشمندی در یکسان‌سازی اسامی و دسته‌بندی‌ها
  - کیفیت تحلیل NLP
  - طراحی و ارزیابی مدل‌های Supervised/Unsupervised و Recommender
- ۱۵٪ – کیفیت کد و ساختار پروژه:
  - تمیزی و ماژولار بودن
  - وجود تست‌ها
  - قابلیت اجرا و تکرارپذیری
- ۱۰٪ – EDA و Insightها:
  - درک اولیه از داده
  - نمودارهای اکتشافی
  - چند Insight جالب و معنادار
- ۱۰٪ – بخش Design Decisions و ملاحظات اخلاقی

---

۹. چند قانون مهم آخر 📌

- **توصیه‌ی خیلی جدی:**
  وارد کردن دستی داده (کپی/پیست تو اکسل و...) ممنوعه 🚫
  کل فرآیند باید خودکار باشه: از خوندن JSON (یا API تلگرام) تا ساخت داشبورد و پیشنهادها.

- **توصیه‌ی جدی:**
  از همون اول روی Git کار کنید.

- **روش کار پیشنهادی (Iterative):**

1. اول مطمئن شید پارسر روی بخش خوبی از داده درست کار می‌کنه.
2. بعد یه داشبورد ساده بسازید که فقط امتیازهای عددی رو نشون بده.
3. بعدش کم‌کم NLP، مدل‌های داده‌کاوی و آخر سر سیستم پیشنهاددهنده رو اضافه کنید.

هدف نهایی این نیست که پیچیده‌ترین مدل دنیا رو بسازین؛
هدف اینه که:

- یه سیستم **کاربردی، پایدار، قابل توضیح و منصفانه** داشته باشین ✅
- و بتونین همه‌ی تصمیم‌های مهم مسیر رو با زبان داده و شواهد توضیح بدین ✅

---

۱۰. زمان‌بندی ⏱️

برای این‌که کار پروژه رو منظم‌تر پیش ببرید، این جدول زمان‌بندی رو رعایت کنید:

فعالیت	تاریخ	توضیحات
مشخص کردن اعضای تیم، تحویل پروپوزال اولیه برای انجام پروژه و ساخت مخزن گیت‌هاب	**جمعه ۱۴ آذر**	مشخص کردن اعضای تیم، آماده‌سازی و بارگذاری پروپوزال اولیه شامل ایده‌ی کلی، ابزارهای پیشنهادی و طرح کلی داشبورد و مدل‌ها، ساخت مخزن گیت‌هاب و دعوت iust-supervisor به مخزن پروژه و پر کردن این فرم
انجام پروژه و تحویل کدها، نتایج و گیت‌هاب پروژه	**جمعه ۱۹ دی**	تحویل نوت‌بوک‌ها، کد اپ/داشبورد، گزارش نهایی و **لینک مخزن گیت‌هاب پروژه**
برگزاری ارائه‌ها (در صورت نیاز)	**زمان متعاقباً اعلام می‌شود**	در صورت نیاز به ارائه‌ی حضوری/آنلاین، تاریخ و جزئیات از طریق کانال اطلاع‌رسانی می‌شود