1. A retail chain wants to forecast product sales using factors like holidays, promotions, location, and inventory. The dataset has 80 features. How should they select the most impactful ones?

Correlation & EDA – Check correlation between sales and features, remove redundant or highly correlated ones.

Statistical Tests – Use methods like Chi-square (for categorical) or ANOVA/F-test (for continuous) to find significant predictors.

Regularization Models – Apply Lasso (L1) regression to shrink less important feature weights to zero.

Tree-based Models – Use Random Forest or XGBoost feature importance to rank features.

Dimensionality Reduction – If many features are correlated, apply PCA to create fewer meaningful components.

2.A podcast platform wants to recommend new podcasts to users based on the categories and hosts they follow. How should they design this content-based system?

Build user profiles using categories and hosts they already follow.

Represent podcasts with features like category tags, host embeddings, and keywords.

Use similarity measures (e.g., cosine similarity, TF-IDF) to match users with similar podcasts.

Rank and recommend the most similar podcasts not yet listened to.

Continuously update user profiles as they follow or listen to new content.

3.An industrial firm wants to predict machine failure using sensor readings and system logs (200+ features). How do they choose which inputs to use?

Start with **EDA and correlation analysis** to drop redundant or low-variance features.

Apply **statistical tests** (ANOVA, Chi-square) to check feature significance with failure labels.

Use **regularization (Lasso/Ridge)** to shrink unimportant features.

Leverage **tree-based models (Random Forest, XGBoost)** for feature importance ranking.

Optionally apply **PCA/autoencoders** to reduce dimensionality while retaining key sensor patterns.

4. A fashion app wants to suggest outfits to users based on their previous likes, color preferences, and seasonal trends. How should they implement the recommendation?

Build a **user profile** capturing liked outfits, preferred colors, and seasonal choices.

Represent outfits with features like style, color palette, fabric, and season tags.

Apply a **content-based filtering** approach to match user profiles with similar outfits.

Blend in **trend data (seasonal/popular items)** to keep recommendations fresh.

Continuously update preferences as users like or dislike new outfits.

5.A research lab wants to predict heart disease using patient data. The dataset has lab results, habits, and vitals. How should they select only the most relevant features?

Perform **EDA & correlation analysis** to remove duplicate or irrelevant features.

Use **statistical tests** (Chi-square for categorical, ANOVA/F-test for numerical) to check significance.

Apply **regularization (Lasso regression)** to shrink weak predictors toward zero.

Use **tree-based models (Random Forest, XGBoost)** for feature importance ranking.

Validate chosen features with **domain knowledge** from doctors for medical relevance.

6. A job portal wants to recommend job listings based on a user's resume and previous applications. How can they build a recommendation system?

Parse the **resume** and extract skills, experience, and job titles.

Represent job listings with similar features (skills, roles, industry, location).

Use **content-based filtering** with similarity measures (TF-IDF, embeddings) to match users to jobs.

Incorporate **collaborative filtering** using patterns from similar applicants' behavior.

Continuously refine recommendations based on user clicks, applications, and saved jobs.

7. A smart home system wants to predict energy usage using readings from multiple sensors. With 100+ features, how do they reduce complexity?

Remove **irrelevant or low-variance features** during preprocessing.

Apply **correlation analysis** to drop highly correlated sensor readings.

Use **feature selection methods** like mutual information, Lasso, or tree-based importance.

Apply **dimensionality reduction (PCA/autoencoders)** to capture key patterns in fewer features.

Validate reduced features by checking prediction accuracy and model interpretability.

8. An e-learning platform wants to suggest tutorials based on a learner's skill gaps identified from quizzes. How should they build this system?

Analyze **quiz results** to detect weak topics and skill gaps.

Tag tutorials with **skills, difficulty level, and prerequisites**.

Use **content-based filtering** to match learners' gaps with tagged tutorials.

Rank tutorials by **relevance and learner's current level**.

Continuously update recommendations as the learner improves through new quizzes.

9.A social media app wants to predict post engagement (likes/comments) using post features, timing, and user activity. How do they select the right features?

Start with **EDA** to identify correlations between engagement and features (e.g., time of posting).

Remove **low-variance or redundant features** that add little value.

Use **statistical tests** (ANOVA, Chi-square) to check significance of categorical/numerical predictors.

Apply **regularization (Lasso)** or **tree-based models (XGBoost, Random Forest)** for feature importance.

Validate results with **A/B testing and domain insights** (e.g., captions, hashtags, active hours).

10. Now you want to create an app named `attendance` in the project and register it.write logic for this Django project

1.Create the project:

django-admin startproject attendance_project

cd attendance_project

2.Create the app named **attendance**:

python manage.py startapp attendance

3.Add `'attendance'` to **INSTALLED_APPS** in `settings.py`.

4.Define models in `attendance/models.py` (e.g., Student, AttendanceRecord).

5.Run `python manage.py makemigrations` and `python manage.py migrate` to register models in the database.