Project #1: Data Science Salary Prediction

Companies worldwide are increasingly relying on data-driven decisions to design competitive salary structures, attract top talent, and retain skilled employees. Compensation for data-focused roles such as Data Scientist, Machine Learning Engineer, and Data Analyst varies significantly based on experience, job role, location, company size, and remote work policies.

In this project, you take the role of a data analyst working with a dataset of data science-related jobs and their corresponding salaries. Your task is to analyze the factors that drive salaries and to build predictive models that can estimate the salary of a data professional in USD based on job and company attributes.

Understanding these salary patterns is valuable for multiple stakeholders. Employers can benchmark their offers more accurately, candidates can better negotiate their compensation, and consulting or HR analytics firms can offer data-driven salary insights to their clients. Instead of relying on guesswork or informal market knowledge, this project uses machine learning models to derive a quantitative relationship between job attributes and salary.

The dataset contains records of job positions across various countries, with features such as work year, experience level, employment type, job title, salary, salary currency, employee residence, remote ratio, company location, and company size. The main goal is to predict the variable **salary_in_usd** using the remaining features.

The dataset (CSV format) can be found here:

https://www.kaggle.com/datasets/arnabchaki/data-science-salaries-2025

The most important fields include (names may vary slightly depending on the dataset version):

- **work_year** – The year the salary was paid
- **experience_level** – Level of experience (e.g., EN, MI, SE, EX)
- **employment_type** – Full-time, part-time, etc.
- **job_title** – Role of the employee (e.g., Data Scientist, ML Engineer)
- **salary_in_usd** – Salary converted to USD (target variable)
- **employee_residence** – Country of the employee
- **remote_ratio** – Percentage of remote work
- **company_location** – Country of the employer
- **company_size** – Company size bucket (e.g., S, M, L)

In this project you will explore, clean, and model this dataset, and finally build a simple user interface that allows a user to input job characteristics and receive an estimated salary.

Tasks:

1. **Explore the data to obtain initial insights.**
   Load the dataset using Pandas. Report the shape of the data (number of rows and columns) and display the first few records. Provide basic information about data types and identify which variables are numerical and which are categorical. Comment briefly on the overall structure of the dataset.

2. **Check and handle missing values and duplicates.**
   Investigate whether the dataset contains any missing values or duplicated rows. If missing values are present, decide and justify a strategy to handle them (e.g., imputation or removal). If duplicates exist, decide whether to keep or remove them and explain your choice.

3. **Produce a descriptive summary of the variables.**
   Generate summary statistics (mean, median, standard deviation, minimum, maximum, quartiles) for the numerical variables, particularly focusing on salary_in_usd. For categorical variables (such as job_title, experience_level, company_size), report frequency counts or proportions. Comment on any notable patterns you observe.

4. **Perform exploratory data analysis with visualizations.**
   Create visualizations to better understand the data and the salary distribution. At a minimum, include:

   - A plot showing the distribution of salary_in_usd
   - Salary_in_usd vs. job_title (you may limit to the most frequent job titles for readability)
   - Salary_in_usd vs. experience_level
   - Salary_in_usd vs. remote_ratio
   - Salary_in_usd vs. company_size
     Additionally, compute and visualize a correlation matrix for the numerical variables (e.g., using a heatmap). Report your main findings and comment on any relationships you notice between features and salary.

5. **Prepare the data for modeling.**
   Select the features you will use as predictors for salary_in_usd. Apply appropriate encoding techniques (such as one-hot encoding) to transform categorical variables into numerical form. Split the dataset into training and test sets using an appropriate proportion (for example, 80% training and 20% testing). For models that require scaling (e.g., Linear Regression and KNN), scale the relevant numerical features and clearly document which transformations are applied.

6. **Build a baseline model using Linear Regression.**
   Fit a Linear Regression model on the training data to predict salary_in_usd. Report the estimated performance on both the training and test sets using R^2, MAE, and RMSE. Discuss briefly how well the baseline model explains the variance in salary and whether there are signs of underfitting or overfitting.

7. **Apply Ridge Regression with hyperparameter tuning.**
   Train a Ridge Regression model to introduce L2 regularization. Use a suitable technique (such as grid search with cross-validation) to tune the regularization parameter alpha. Compare the performance of the tuned Ridge model to the baseline Linear Regression in terms of R^2, MAE, and RMSE on the test set. Comment on any improvement or changes in model behavior.

8. **Train a KNN Regression model and tune the number of neighbors.**
   Build a K-Nearest Neighbors Regressor to model the potentially non-linear relationship between features and salary. Use GridSearchCV or a similar method to tune the number of neighbors (K) and any other relevant hyperparameters. Evaluate the best KNN model using R^2, MAE, and RMSE, and compare its performance to the linear models.

9. **Train a Decision Tree Regressor and analyze feature importance.**
   Fit a Decision Tree Regressor on the training data. Tune key hyperparameters such as maximum depth and minimum samples required to split a node. After training, extract and report feature importance values from the tree model. Evaluate the Decision Tree's performance on the test set and compare it to the previous models.

10. **Train a Random Forest Regressor and identify the best-performing model.**
    Apply a Random Forest Regressor to the same problem. Use hyperparameter tuning (for example, varying the number of trees, maximum depth, and maximum features) to identify a performant configuration. Evaluate the Random Forest model using R^2, MAE, and RMSE on the test set. Summarize the performance of all five models (Linear Regression, Ridge Regression, KNN, Decision Tree, Random Forest) in a comparison table and identify which model performs best according to your chosen criteria.

11. **Save the final model and preprocessing pipeline.**
    Once the best model has been selected, save it to disk using a suitable serialization method (such as pickle or joblib). Also save the preprocessing steps (such as encoders and scalers) so that new data can be transformed and passed to the model in a consistent way. Document how someone can reload the model and pipeline to make predictions later.

12. **Develop a simple Streamlit user interface for salary prediction.**
    Build a basic Streamlit application that allows a user to input key job attributes (for example, job_title, experience_level, remote_ratio, company_size, and other relevant fields). The app should apply the same preprocessing steps, pass the transformed input to the saved model, and display the predicted salary_in_usd. Optionally, include a short explanation or summary of the prediction result.

13. **Summarize your findings and provide conclusions.**
    Write a brief conclusion section summarizing what you have learned about the factors that influence data science salaries in this dataset. Highlight which features appear most important, how well your best model performs, and how this work could be extended (for example, by including more features, trying boosting algorithms, or deploying the Streamlit app in production).