Technical Challenge: Data Scientist

Develop a predictive model (using Python) to forecast an individual's salary based on a given dataset. The code should be hosted in a **Github as a public repo**, and the dataset should be included in the repository or accessible via a provided link.

Source data:

- You will be sent a dataset. Use this dataset to develop your predictive model. Feel free to engineer additional features to enhance the predictive power of your model.
- Context: the data corresponds to a public dataset, and it includes information about their age, gender, education level, job title, years of experience and description. Using this data, it should be possible to predict (approximately) an employee's salary based on these variables.

You will be given a CSV file containing the data, and you should write the code to read from that file. Moreover, you should design and implement a predictive model to forecast salaries, including all necessary preprocessing steps, feature engineering, and model evaluation logic. The model should be trained and tested on the provided dataset.

Before the technical interview, the candidate has to provide a URL for the public Github repository, and the dataset should be included in the repo. The repo's README should contain relevant context and necessary information to understand the predictive model and the steps taken during the development process. Github is not meant to be used as a hosting solution but as source control, meaning that a clean commit history reflecting the progress is expected.

Clarification: the usage of GenAI tools (e.g. ChatGPT, Cursor, TabNine, Github Copilot) is allowed and it is expected to streamline repetitive and/or easy to automate processes such as the following:

- Generate descriptive git commit messages.
- Generate docstrings for functions.
- Generate drafts for tests and boilerplate code.
- Use as a peer review tool to identify potential flaws.
- Identify which design patters will be most suitable for a particular functionality

Feel free to experiment and leverage these tools to enhance your work.

Mandatory Features:

- Load and preprocess the dataset using Pandas / Polars or similar tools.
- Perform feature transformations on the input dataset to derive useful numerical features from the model.
- Develop a predictive model using a machine learning library such as Scikit-learn or TensorFlow/Keras.
- Evaluate the model's performance using appropriate metrics. Explaining why that metric was useful.

- Compare the model results against a sensible baseline – e.g. a DummyRegressor
- All the metrics should be reported as intervals, not as single estimates. See references on how to calculate confidence intervals for ML.
- Structure the code in modules . See the reference to know what is expected.
- Present the final result in a Jupyter Notebook which should:
    - Have a Table of Contents at the top.
    - Contain explanatory text in markdown, properly formatted.
    - Have no big blocks of code, instead it should import the functionality from the modules and only have initial configuration.
    - Include illustrations or images showing the results.

Optional Features (include at least 1):

1. Explain the assumptions of the model used and validate if they are met.
2. Perform hyperparameter tuning to optimize the model's performance – e.g. Optuna.
3. Validate the model using cross-validation techniques.
4. Visualize the relationships between features and the target variable.
5. Use an interpretation tool to explain the contribution of each feature - e.g. SHAP
6. Use a more advanced model such as Random Forest, or Neural Networks.
7. Use multiple models and combine their outputs (e.g. Bagging, Boosting, Voting).
8. Lock your dependencies - e.g., using pipenv, uv or pdm.
9. Create a simple REST API for salary predictions - e.g. FastAPI.
10. Create a simple UI for salary predictions - e.g. Streamlit, Gradio, Taipy.
11. Save model predictions in a SQL Database - e.g. SQLite.
12. Use an Experiment Tracking tool - e.g. Weights and Biases, Neptune, MLFlow, Tensorboard.
13. Implement a Drift Detection technique and simulate a failure - e.g. Frouros.
14. Use version control for datasets - e.g. DVC.
15. Create anonymized datasets based on the given dataset - e.g. Synthetic Data Vault (SDV).
16. Use an interactive data visualization tool - e.g. Altair.
17. Add Unit tests for the project – e.g. Pytest.
18. Use typed dataframes – e.g. Pandera
19. Use calibration techniques in the model validation.

Evaluation criteria:

The project will be evaluated in five dimensions:

- **Model design**: how the predictive model is structured, the choice of features, model parameters, etc.
- **Best practices**: a code review to evaluate best practices, patterns, and overall design.
- **Complexity**: the level of optional features implemented and how they were incorporated into the development process.
- **Organization**: the most important aspect is to have something functional by the deadline; the candidate should be able to organize themselves and deliver something valuable.
- **Innovation**: new features outside of what has been asked are a plus.

Do **NOT** focus too much on having the best model, although the model should be used to predict salary, the approach will be evaluated over the actual performance of the model.

References

Here are some useful links and tutorials to assist you in completing the technical challenge:

Data Preprocessing and Feature Engineering

1. **Pandas Documentation and tutorials**:
    a. Marc Garcia: Introducción a pandas (español) | PyData Córdoba 2019
    b. Matt Harrison: Effective Pandas | PyData Salt Lake City Meetup 2021
    c. Daniel Chen: Introduction to Data Processing in Python with Pandas | SciPy 2019 Tutorial
    d. Daniel Chen: Cleaning and Tidying Data in Pandas | PyData DC 2018
    e. James Powell: So You Wanna Be a Pandas Expert? (Tutorial) | PyData Global 2021
2. **Polars Documentation**:
    a. Matt Harrison: Getting Started with Polars | PyCon 2023
    b. Matt Harrison: An Introduction to Pandas 2, Polars, and DuckDB | PyData Global 2023
3. **Feature Engineering Techniques**:
    a. Feature Engineering for Machine Learning | Google
    b. Kaggle Feature Engineering Course
    c. Vincent Warmerdam: Winning with Simple, even Linear, Models | PyData London 2018
    d. Kajanan Sangaralingam and Anindya Datta: Feature Engineering Made Simple | PyData London 2022
    e. Franziska Horn: Automated Feature Engineering and Selection in Python | PyData Berlin 2019
    f. Thorben Jensen: Automating feature engineering for supervised learning? methods, open-source tools and prospects | PyData Berlin 2019

Predictive Modeling

1. **Scikit-learn Documentation**:
    a. Scikit-learn Documentation
    b. Scikit-learn Tutorials
2. **TensorFlow/Keras Documentation**:
    a. TensorFlow Getting Started
    b. Keras Documentation

Model Evaluation and Metrics

1. **Model Evaluation Metrics**:
    a. Evaluation Metrics in Scikit-learn
    b. Jason Brownlee: How to Choose Evaluation Metrics | Machine Learning Mastery
2. **Confidence Intervals for Machine Learning**:
    a. Sebastian Raschka: Creating Confidence Intervals for Machine Learning Classifiers
    b. Jason Brownlee: How to Calculate Confidence Intervals | Machine Learning Mastery

Code Structuring and Best Practices

1. **Modular Code Design**:
    a. Arjan Codes: Restructuring a Data Science Project

   b. Moussa Taifi: Clean Machine Learning Code: Practical Software Engineering Principles for ML Craftsmanship | PyData New York 2019
   c. Software **Engineering** for Data Science | Skoltech
2. **Kedro**
   a. Jose Miguel Nuñez Darnott: Kedro: un framework de Python para crear código DS mantenible y modular | PyCon Chile 2022
   b. Datta & Rodríguez - Building the composable Python data stack with Kedro & Ibis | PyData London 2024
   c. Yetunde Dada: Production-level data pipelines that make everyone happy using Kedro | PyData Austin 2019
   d. Carlos Gimenez: Data Science Best Practices con Kedro | PyData Córdoba 2019
   e. Ionut Barbu & Tim Brakenhoff - How to build production-ready data science pipelines with Kedro | PyData Eindhoven 2023

Advanced Techniques and Optional Features

1. **Hyperparameter Tuning**:
   a. Hyperparameter Tuning with Scikit-learn
   b. Hyperparameter Tuning with Keras
   c. Shotaro Sano: Optuna: A Define by Run Hyperparameter Optimization Framework | SciPy 2019 |
   d. Vincent Warmerdam: Optuna: a hyperparameter optimization framework | Probabl
2. **Cross-Validation Techniques**:
   a. Cross-Validation in Scikit-learn
   b. Prashant Gupta: Cross-Validation Explained | Towards Data Science
   c. Sergey Feldman: You Should Probably Be Doing Nested Cross-Validation | PyData Miami 2019
   d. Jan van der Vegt: Common pitfalls leading to wrongly estimated model performance | PD Eindhoven 2019
   e. Benjamin Bengfort: Visual Diagnostics for More Effective Machine Learning | PyData Miami 2019
   f. Maria Khalusova: Machine Learning Model Evaluation Metrics | PyData LA 2019
3. **Feature Importance and SHAP**:
   a. Alexis Cook: Understanding SHAP Values | Kaggle
   b. Kevin Lemagnen: Open the Black Box: an Introduction to Model Interpretability with LIME and SHAP | PyData New York 2018
   c. Adi Watzman: SHAP Values for ML Explainability | PyData Tel Aviv Meetup
   d. Layla Yang: Unified Approach to Interpret Machine Learning Model SHAP + LIME - Databricks
   e. Samuel Jenkins and Harsha Nori: How to Explain Models with IntepretML Deep Dive | Microsoft
   f. Ben Fowler: Traditional & Novel Feature Selection Approaches | PyData LA 2019
   g. Hidde Hovenkamp: SHAP and Beyond | PyData Amsterdam 2019
4. **Experiment Tracking Tools**:
   a. Using Weights and Biases

    b. Tobias Sterbak: Managing the end-to-end machine learning lifecycle with MLFlow | PyData Berlin 2019
    c. Vladimir Osin, Milan Mulji: Managing Machine Learning Lifecycle with MLflow | PyData Eindhoven 2019
    d. Tom Goldenberg: Kedro + MLflow - Reproducible and Versioned data pipelines at scale | PyData LA 2019
    e. Machine Learning Lifecycle Made Easy with MLflow | PyData Global 2021
    f. Tobias Sterbak: Introduction to MLOps with MLflow | PyData Berlin 2022
5. **Interactive Data Visualization**:
    a. Jake VanderPlas - Exploratory Data Visualization with Vega, Vega-Lite, and Altair - PyCon 2018
6. **Creating REST APIs**:
    a. Sebastián Ramírez: Building REST APIs with FastAPI | Real Python
7. **Building Simple UIs**:
    a. Lisa Carpenter: How to create beautiful interactive GUIs and web apps | PyCon 2023
    b. Joe Cheng - Shiny for Python: Interactive apps and dashboards made easy-ish | PyData NYC 2022
    c. Furkan M. Torun - Become a Data Storyteller with Streamlit! | PyData Prague 2023
    d. Thomas J. Fan: Light Up Your Data with Streamlit | SciPy 2021
    e. Caroline Frasca, Tony Kipkemboi: Advanced Streamlit for Python Developers | PyCon 2024
8. **Unit Testing**:
    a. Anthony Shaw: Unit Testing in Python | Real Python
    b. Jes Ford - Getting Started Testing in Data Science - PyCon 2019
    c. Zac Hatfield-Dodds - Escape from auto-manual testing with Hypothesis! - PyCon 2019
9. **Typed DataFrames**:
    a. Niels Bantilan: Pandera: Statistical Data Validation of Pandas Dataframes | SciPy 2020
    b. Niels Bantilan - Pandera: Beyond Pandas Data Validation | SciPy 2023
    c. Nathan McDougall: Robust Data Workflows Made Easy: Classes with Pandera and Pydantic | KiwiPyCon 2023
    d. Niels Bantilan: Statistical Typing: A Runtime TypingSystem for Data Science&Machine Learning | PyCon US 2021
    e. Niels Bantilan: Robust, End-to-end Online ML Applications with Flyte, Pandera and Streamlit | PyData Global 2021
    f. Natan Mish - Data Validation for Data Science | PyData 2022
    g. Matt Harrison - Testing Pandas- Shoots, leaves, and garbage! | PyData Global 2022
    h. Theodore Meynard - Test your data like you test your code | Pydata London 2022
10. **Version Control:**
    a. Dean Pleban: DVC Showcase – Who Moved My Data? | PyData Global 2021
    b. Talks - Rob de Wit: Transforming a Jupyter Notebook into a reproducible pipeline for ML experiments
    c. Antoine Toubhans: Flexible ML Experiment Tracking System for Python Coders with DVC and Streamlit | PyData Berlin 2022
    d. Estefania Barreto-Ojeda: Applications in ML Drug Discovery pipelines | PyData NYC 2022
11. **Data Control**

a. Cesar Garcia: Improving Open Data Quality using Python | PyData Global 2023
b. Moritz Meister: Data Validation for Feature pipelines: Using Great Expectations and Hopsworks | PyData Global 2022
c. Theodore Meynard: Test your data like you test your code | PyData London 2022
d. Beatriz Blanco: Tutorial Great Expectations.

12. Anonymize Data

a. Antonia Scherz: Unlocking Information - Creating Synthetic Data for Open Access.
b. Gatha: Do I need to be Dr. Frankenstein to create real-ish synthetic data?
c. Aileen Nielsen: An Overview to Simulations and Generating Synthetic Data Sets | Scipy 2019 Tutorial

13. Calibration

a. Inbar Naor: Model Calibration - is your model ready for the real world? - PyCon Israel 2018
b. Samuel Rochette: Quantifying uncertainty in machine learning models | PyData New York 2019
c. Gordon Chen: Safe Handling Instructions for Probabilistic Classification | SciPy 2019
d. Maria Navarro: Quantifying uncertainty in Machine Learning predictions | PyData London 2019

14. Assumptions

a. Jonas Kristoffer Lindeløv: Common statistical tests are linear models (or: how to teach stats)
b. Matheus Facure: The Unreasonable Effectiveness of Linear Regression