**Task 1: Exploratory Data Analysis**

**Exploratory Data Analysis** is the first step in understanding your data and acquiring domain knowledge. It helps in knowing the real features apart. Perform EDA using the following resources and note down your insights with the most visualizations that you perform.

* AutoEDA Libraries:

[dabl](https://amueller.github.io/dabl/dev/index.html)

[AutoViz](https://github.com/AutoViML/AutoViz)

📍 [code](https://drive.google.com/drive/folders/1bnYEkAwdt6Rn3hrbIUsnHJRCg5uoPElu)

* Visualization with python

[All conventional libraries](https://towardsdatascience.com/introduction-to-data-visualization-in-python-89a54c97fbed)

[Understanding Matplotlib](https://towardsdatascience.com/data-visualization-using-matplotlib-16f1aae5ce70)

📍 [code](https://colab.research.google.com/drive/1SUW8kOeiPxcA3y7XdvM4hUd6hTmiuqF9?usp=sharing)

[EDA at multiple dimensions](https://towardsdatascience.com/the-art-of-effective-visualization-of-multi-dimensional-data-6c7202990c57) ***(only for intermediate/advance level)***

* Techniques for categorical data ***(relevant to this dataset)***

[13 ways to visualize categorical data with 📍 code](https://towardsdatascience.com/13-key-code-blocks-for-eda-classification-task-94890622be57)

**Task 2: Preprocessing**

**Data preprocessing** is a proven method for resolving issues with incomplete, noisy, and inconsistent raw input data. Issues such as ambiguous and unexpected results of modeling on raw data. With the insights of EDA, you must've understood the shortcomings of your dataset. Use preprocessing techniques to clean and remove those vulnerabilities from the dataset.

***(Note: only techniques marked [✔] are relevant to the provided dataset)***

* [Missing value treatment](https://www.kaggle.com/code/parulpandey/a-guide-to-handling-missing-values-in-python/notebook) ✔
* [Encoding techniques](https://towardsdatascience.com/all-about-categorical-variable-encoding-305f3361fd02) ✔ *(Note: We would use Label Encoder or One Hot encoder in our case)*
* [Imbalanced data treatment](https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook) ✔
* [Feature Scaling](https://www.analyticsvidhya.com/blog/2021/05/feature-scaling-techniques-in-python-a-complete-guide/) *(for numerical features, if needed)*
* [Skewed data treatment](https://opendatascience.com/transforming-skewed-data-for-machine-learning/)
* [Outlier detection and treatment](https://towardsdatascience.com/ways-to-detect-and-remove-the-outliers-404d16608dba)
* [Feature Scaling](https://www.analyticsvidhya.com/blog/2021/05/feature-scaling-techniques-in-python-a-complete-guide/) *(for numerical features, if needed)*
* [Dimensionality reduction techniques](https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b) *(for large feature dataset)*

**Task 3: Feature Engineering and Modeling**

**Feature Engineering** is an encapsulated process of feature selection, extraction, transformation and elimination. When the dataset has uncertain columns that can cause problems in the modeling process, like bulky models with a low score, higher time complexities, and unnecessarily complex models. After preprocessing, feature engineering techniques are carried out to select only helpful features.

* [Feature Extraction fundamentals](https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114)
  + [Resampling strategies for Imbalanced Data](https://www.kaggle.com/code/rafjaa/resampling-strategies-for-imbalanced-datasets/notebook)
* [How to select feature selection technique for your data?](https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/)
  + [Feature Selection for categorical data](https://machinelearningmastery.com/feature-selection-with-categorical-data/)

**Modeling** in machine learning is creating a mathematical representation by generalizing and learning from training data. Then, the built machine learning model is applied to new data to make predictions and obtain results. Below are some beginner-friendly resources to learn all about ML models.

* [Exploring cross-validation methods](https://towardsdatascience.com/train-test-split-and-cross-validation-in-python-80b61beca4b6)
* [What is an ML model?](https://towardsai.net/p/machine-learning/what-is-an-ml-model)
* [Step-wise guide to build a baseline model](https://analyticsindiamag.com/step-by-step-building-block-for-machine-learning-models/) (classification)
* [End-to-end multiclass classification model](https://medium.com/analytics-vidhya/building-classification-model-with-python-9bdfc13faa4b) (with 📍code)

**Model Evaluation** is equivalent to quality assurance for your machine learning model. You evaluate your machine learning model using matrix calculations, quality measurements, and the model metric approach. It is done by a statistical metric decided beforehand (based on the type of dataset). To learn about evaluation metrics, and their application in model evaluation, follow the below resource in order.

***(Note: the metric for the given dataset is*** ***f1-score)***

* [Learning all about evaluation metrics](https://towardsdatascience.com/20-popular-machine-learning-metrics-part-1-classification-regression-evaluation-metrics-1ca3e282a2ce)
* [Metrics related to classification problems](https://towardsdatascience.com/understanding-confusion-matrix-precision-recall-and-f1-score-8061c9270011#:~:text=Precision%2DRecall%20Trade%2DOff)
* [Multi-class classification metrics](https://towardsdatascience.com/multi-class-metrics-made-simple-part-ii-the-f1-score-ebe8b2c2ca1)
* [StackExchange Thread: Weighted f1-score](https://stats.stackexchange.com/questions/463224/which-performance-metrics-for-highly-imbalanced-multiclass-dataset)

**Task 4: Hyperparameter Tuning**

**Model Tuning** is a process where we define search space algorithms that can optimize the hyperparameters of a baseline model. Hyperparameters are manual parameters that can impact a model's learning capacity. Optimization of those parameters is called hyperparameter tuning.

***(Note: You can use either one of the following techniques to get the desired results.)***

* [Using GridSearchCV](https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/)
* [Tuning faster with GridSearchCV](https://towardsdatascience.com/faster-hyperparameter-tuning-with-scikit-learn-71aa76d06f12)
* [Using RandomisedSearchCV](https://analyticsindiamag.com/guide-to-hyperparameters-tuning-using-gridsearchcv-and-randomizedsearchcv/) 📍 [code](https://machinelearningmastery.com/hyperparameter-optimization-with-random-search-and-grid-search/)
* [Using Optuna](https://towardsdatascience.com/tuning-hyperparameters-with-optuna-af342facc549) 📍[code](https://www.kaggle.com/code/saurabhshahane/catboost-hyperparameter-tuning-with-optuna)
* [Hyperopt](https://machinelearningmastery.com/hyperopt-for-automated-machine-learning-with-scikit-learn/)

**Task 5: Explainable AI**

*Note: use SHAP version* ***0.38.0*** *to* ***0.40.0***

Documentation: [Click Here](https://shap.readthedocs.io/en/latest/index.html)

[Explainable AI (XAI) with SHAP -Multi-Class Classification Problem](https://towardsdatascience.com/explainable-ai-xai-with-shap-multi-class-classification-problem-64dd30f97cea) 📊

[The SHAP with More Elegant Charts](https://medium.com/dataman-in-ai/the-shap-with-more-elegant-charts-bc3e73fa1c0c) ⭐

[How to Explain your models using LIME](https://medium.com/dataman-in-ai/explain-your-model-with-lime-5a1a5867b423) 🍋

📋 Research Papers: [LIME](https://arxiv.org/pdf/1602.04938.pdf) | [SHAP](https://proceedings.neurips.cc/paper_files/paper/2017/file/8a20a8621978632d76c43dfd28b67767-Paper.pdf)

📺 Presentation slides: [Click Here](https://www.canva.com/design/DAFSe2pHy6Y/BPD-m4nfBvDuTm04COFpKQ/view?utm_content=DAFSe2pHy6Y&utm_campaign=designshare&utm_medium=link&utm_source=publishsharelink)

📚Reading guides (for theoretical understanding)

[Force plot](https://github.com/slundberg/shap/issues/977)

[Decision plot](https://towardsdatascience.com/introducing-shap-decision-plots-52ed3b4a1cba)

📍 [code](https://slundberg.github.io/shap/notebooks/plots/decision_plot.html)

[Heatmap](https://shap.readthedocs.io/en/latest/example_notebooks/api_examples/plots/heatmap.html)

[Shap Values](https://towardsdatascience.com/shap-explained-the-way-i-wish-someone-explained-it-to-me-ab81cc69ef30)

[Limitations](https://towardsdatascience.com/the-limitations-of-shap-703f34061d86) | [Limitations 2.0](https://towardsdatascience.com/using-shap-for-explainability-understand-these-limitations-first-1bed91c9d21)

**Task 6: Deployment**

**Deployment with Streamlit**

1. Download streamlit in your environment using pip.
2. Prepare python files, to process the user data and return prediction, and one for the streamlit code for the Front End
3. Prepare files that would be useful for deploying on [Render](render.com): <setup.sh>, requirements.txt

📌 Creating ***"requirements.txt"***:

* + GoTo your project folder and open command prompt in that folder.
  + install pipreqs package using "pip install pipreqs"
  + then type this command: for requirements.txt: pipreqs --encoding=utf8 --debug "path/to/project". *Don't forget to add absolute path above inside inverted commas*

1. Make a new repository on GitHub and push the project folder containing all of the folders into the repository

📌 Guide to adding files on ***GitHub***:

* + <https://docs.github.com/en/github/managing-files-in-a-repository/managing-files-on-github/adding-a-file-to-a-repository>
  + video: <https://www.youtube.com/watch?v=vbQ2bYHxxEA&ab_channel=StudyZone>

1. Goto Render and Log-in. Create a new app by selecting **Web Service** and give it a name
2. In the "Deploy" section of this new app, choose "GitHub"
3. in the next section, search for your app by the exact name, and click on "connect" to connect the app
4. Once connected, write "pip install -r requirements.txt" as the Build Command
5. Write "sh <setup.sh> && streamlit run <app.py>" as Start Command, then click on "Advanced"
6. In the Advanced drop-down menu, Select "No" for the Auto Deploy option. Then click on "Create Web Service"
7. if the above said procedures are followed, the build logs will show your deployed URL at the end

👜 **SAMPLE REPOSITORY:** [Click Here](https://github.com/saurabhshahane4/RTA-Project)

🧩 **REFERENCE BLOG FOR COMPLETE PROCEDURE:** [Click Here](https://medium.com/geekculture/lets-build-and-deploy-your-first-machine-learning-app-fa350ec6b5cf)

**Deployment with Flask**

1. Create environment using python -m venv *<name\_of\_env>*. Install packages for the project with the requirements.txt file given here.

pip install -r requirements.txt

1. Write your application with necessary python files. Add front-end files according to your requirement.
2. Flask app can be deployed as an API or as a WebApp. API doesn't need a front-end architecture (refer to recording for more info).
3. Once your application is ready, add requirements.txt with the packages you used. Use the command pip freeze > requirements.txt to update/create the file.
4. Upload the project folder (contents) to GitHub. Login to a hosting service of your choice and create a new Web Service.

Suggestions: <www.render.com> OR <www.railway.app>

1. Connect your GitHub repository to the service. Make sure to give your app a name, the start command, and choose the branch you mean to deploy. Turn auto-deploy to "Off".
2. Click on "Deploy" at the bottom and you will see the build logs.
3. Once the deployment is complete, you can access your application through the link given on top, under the title of the app.

👜 **SAMPLE REPOSITORY:** [Click Here](https://github.com/visalakshi2001/wild-blueberry-flask-app)

**MLFlow**

1. 📻 Presentation: [Click here](https://gamma.app/public/MLflow-xay7c95jeddtprp)
2. 👜 **SAMPLE REPOSITORY:** [Click Here](https://github.com/ChiragChauhan4579/MLflow)
3. MLFlow Website: <mlflow.org>