Screening Test:

As part of the screening test, you will write code to parse the JSON file provided(algoparams_from_ui) and kick off in sequence the following machine learning steps programmatically. Keep in mind your final code should be able to parse any Json that follows this format. It is crucial you have a generic parse that can read the various steps like feature handling, feature generation and model building using Grid search after parsing hyper params.

1) Read the target and type of regression to be run

```
"target": {
    "prediction_type": "Regression",
    "target": "petal_width",
    "type":"regression",
    "partitioning": true
},
```

2) Read the features (which are column names in the csv) and figure out what missing imputation needs to be applied and apply that to the columns loaded in a dataframe

```
"feature_handling": {
  "sepal_length": {
    "feature_name": "sepal_length",
    "is_selected": true,
    "feature_variable_type": "numerical",
    "feature_details": {
      "numerical_handling": "Keep as regular numerical feature",
      "rescaling": "No rescaling",
      "make_derived_feats": false,
      "missing_values": "Impute",
      "impute_with": "Average of values",
      "impute_value": 0
    }
  },
```

3) Compute feature reduction based on input. See the screenshot below where there can be No Reduction, Corr with Target, Tree-based, PCA. Please make sure you write code so that all options can work. If we rerun your code with a different Json it should work if we switch No Reduction to say PCA.

```
"feature_reduction": {
  "feature_reduction_method": "Correlation with target",
  "No Reduction": {
    "is_selected": true,
    "num_of_features_to_keep": 5
  },
  "Correlation with target": {
    "is_selected": false,
    "num_of_features_to_keep": 0
  },
  "Tree-based": {
    "is_selected": false,
    "num_of_features_to_keep": 0,
    "depth_of_trees": 0,
    "num_of_trees": 0
  },
  "Principal Component Analysis": {
    "is_selected": false,
    "num_of_features_to_keep": 0
  },
```

4) Parse the Json and make the model objects (using sklean) that can handle what is required in the “prediction_type” specified in the JSON (See #1 where “prediction_type” is specified). Keep in mind not to pick models that don’t apply for the prediction_type specified

```
"LogisticRegression": {
  "model_name": "LogisticRegression",
  "is_selected": false,
  "parallelism": 2,
  "min_iter":30,
  "max_iter":50,
  "min_regparam":0.5,
  "max_regparam":0.8,
  "min_elasticnet":0.5,
  "max_elasticnet":0.8
},
```

5) Run the fit and predict on each model – keep in mind that you need to do hyper parameter tuning i.e., use GridSearchCV

6) Log to the console the standard model metrics that apply

Note:

1. **PLEASE NOTE THAT, WE HAVE A ZERO TOLERANCE POLICY FOR PLAGIARISM. IF YOU PLAGIARIZE THE TEST, YOU WILL BE CAUGHT AND IMMEDIATELY TERMINATED.**
2. **Please do not submit the code if code is not up to a standard.**
3. **Please do not send LinkedIn Request to Connect!**
4. **PLEASE MAKE SURE YOU SUBMIT EVERYTHING VIA A GITHUB LINK AND PLS UPLOAD ALL ASSETS AND FILES.**