UM HACKATHON 2022

Aging Days Prediction and Classification

CHEONG YI FONG

CHEW YAO DONG

CHIEW ZHE WEI

LEE KAI YANG

MITCHEL KEAGAN THESEIRA

Prepared by: Team Donald Duck

PRESENTATION OUTLINE

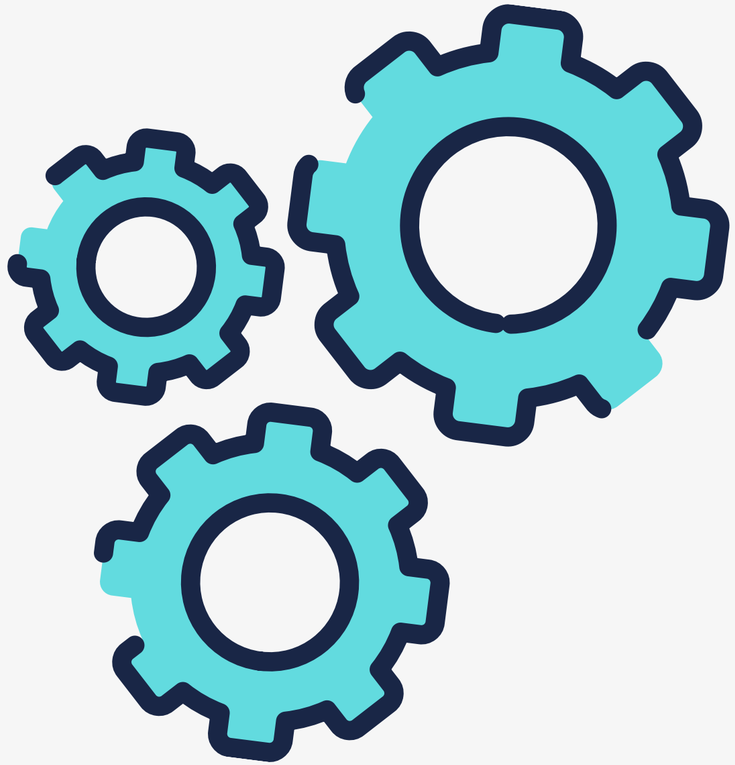
1. INTRODUCTION
2. TECHNICAL IMPLEMENTATION
3. PREDICTING BATTERY AGING
4. CATEGORIZING PRIORITIES
5. BUSINESS VALUES

INTRODUCTION

PROBLEM STATEMENT

Base transceiver station (BTS) locations, technical operation centres, and core network components are all scheduled for maintenance.

Standard guidelines provide on maintenance once or twice a year.



Depending on the predicted equipment lifespan and failure, plan maintenance.

Maintain equipment as needed and to avoid any failures rather than just fixing them when they occur.

OBJECTIVES

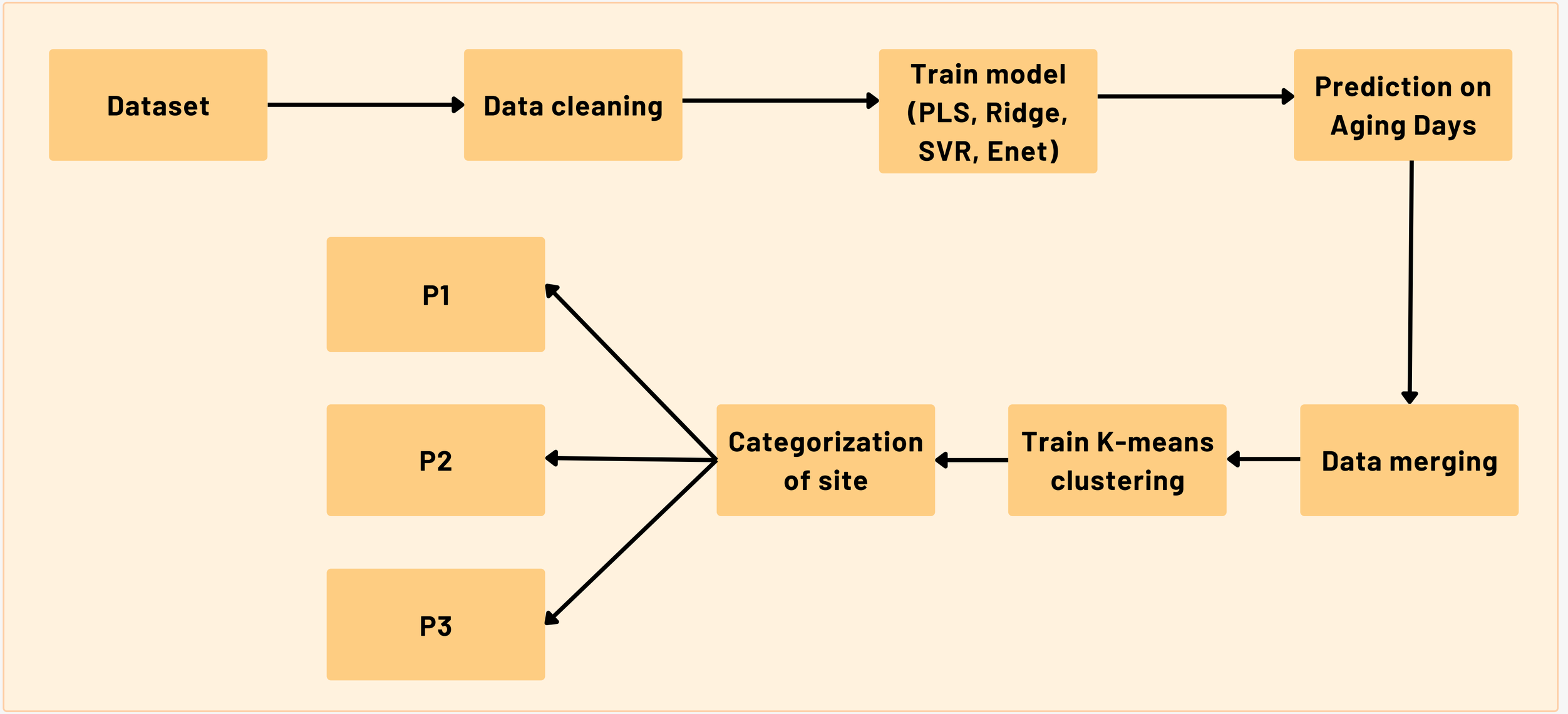
Problem 1

To predict the aging of old and new equipment and lifespan of the backup batteries based on the batteries' specifications and environmental factors using prediction model

Problem 2

To categorize and classify sites into priorities P1, P2, and P3 depending on the predicted battery aging and the status of the existing gensets on site.

Model Lifecycle

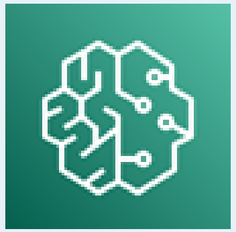


TECHNICAL IMPLEMENTATION

TECHNICAL IMPLEMENTATIONS

AWS Data Wrangler

Amazon Simple
Storage Service (S3)



Amazon SageMaker

AWS Data Wrangler

01 02 03

Feature Engineering

Data Cleaning

Encoding

SMOTE

Data Analysis

Bias Report

Multicollinear

Feature Correlation

Data Visualization

Histogram

Quality and Insights

Table Summary

AWS Data Wrangler (cont.)

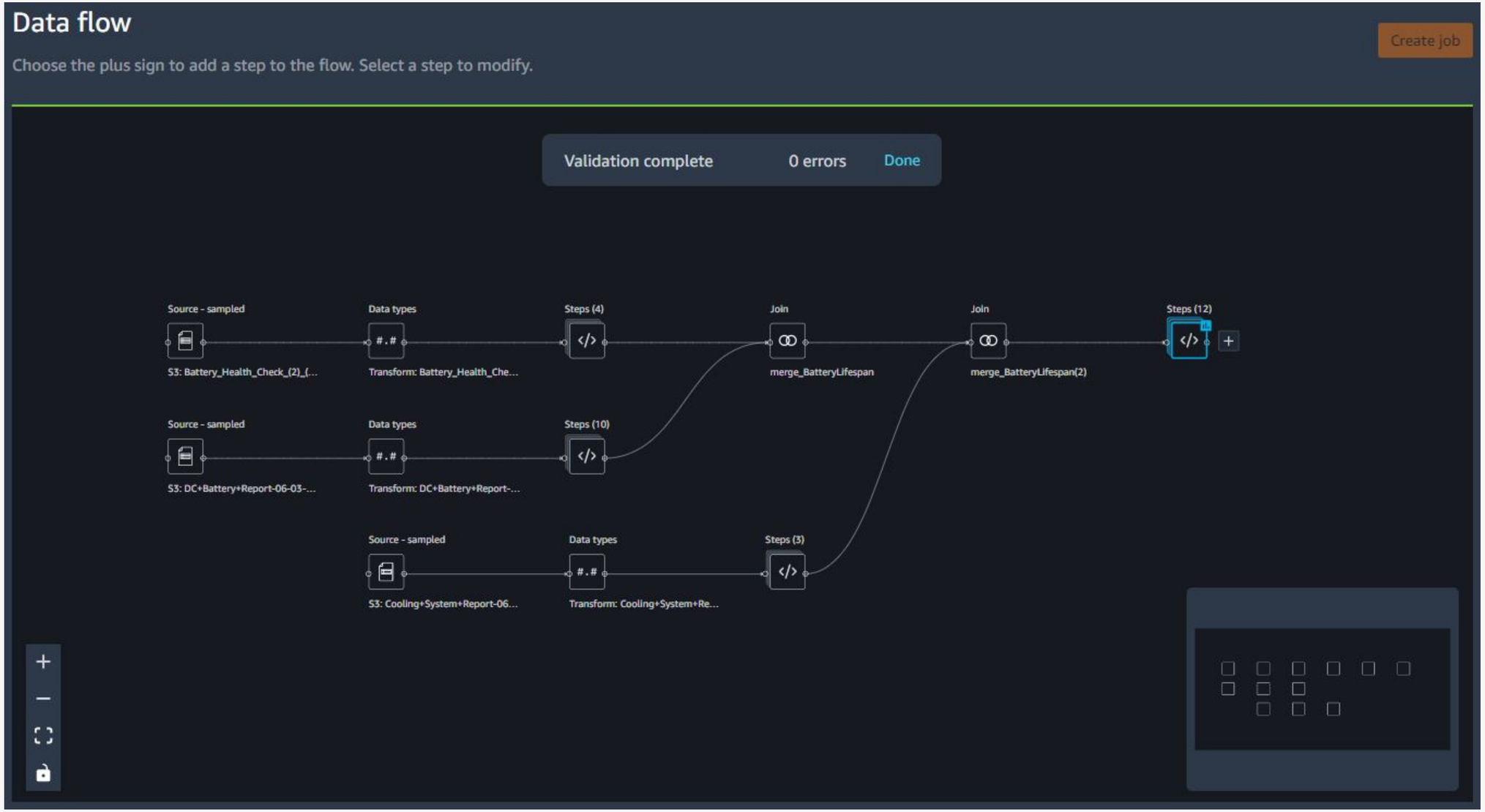
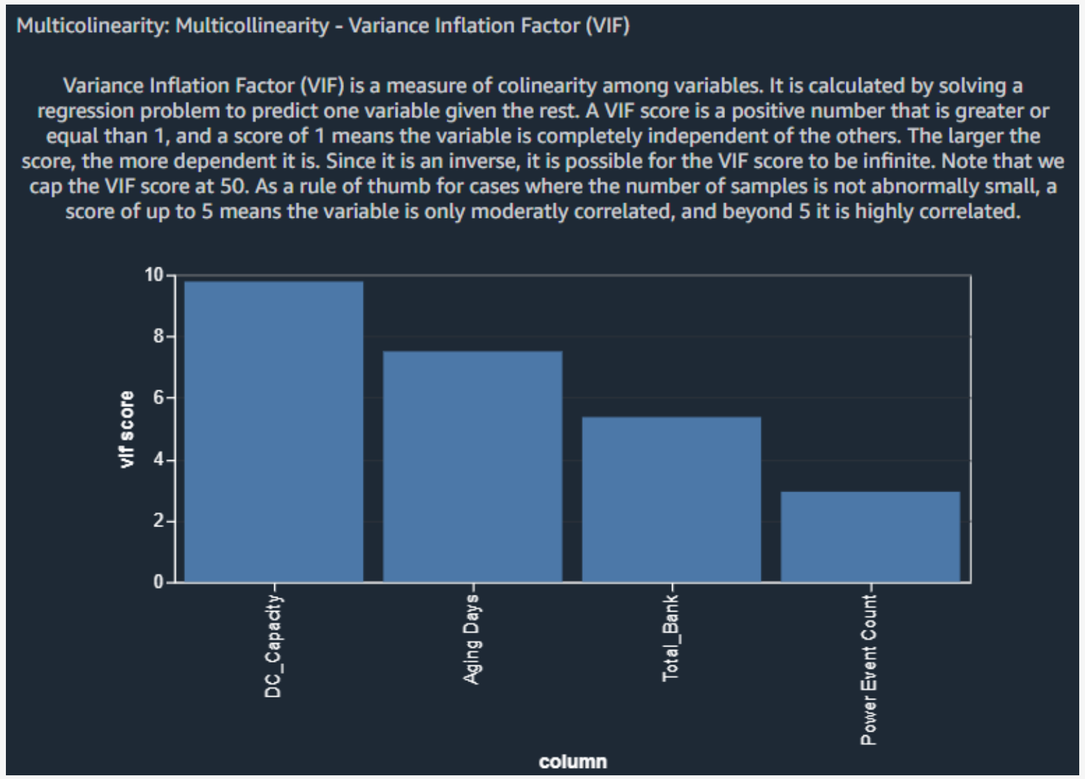


Figure 1.0: Data flow for preprocessing

Data Wrangler Analysis

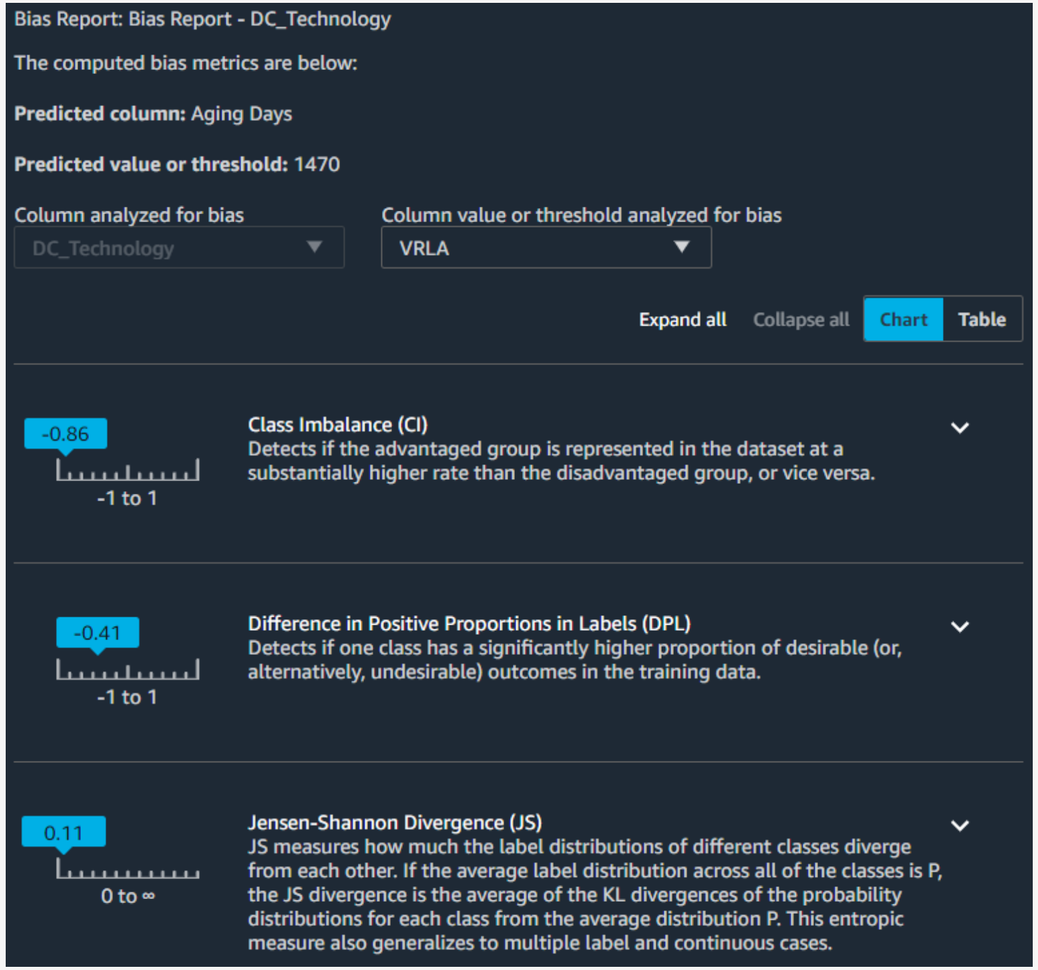
Multicollinearity



Multicollinearity is a statistical concept where several independent variables in a model are correlated and is calculated by VIF score (Variance Inflation Factor).

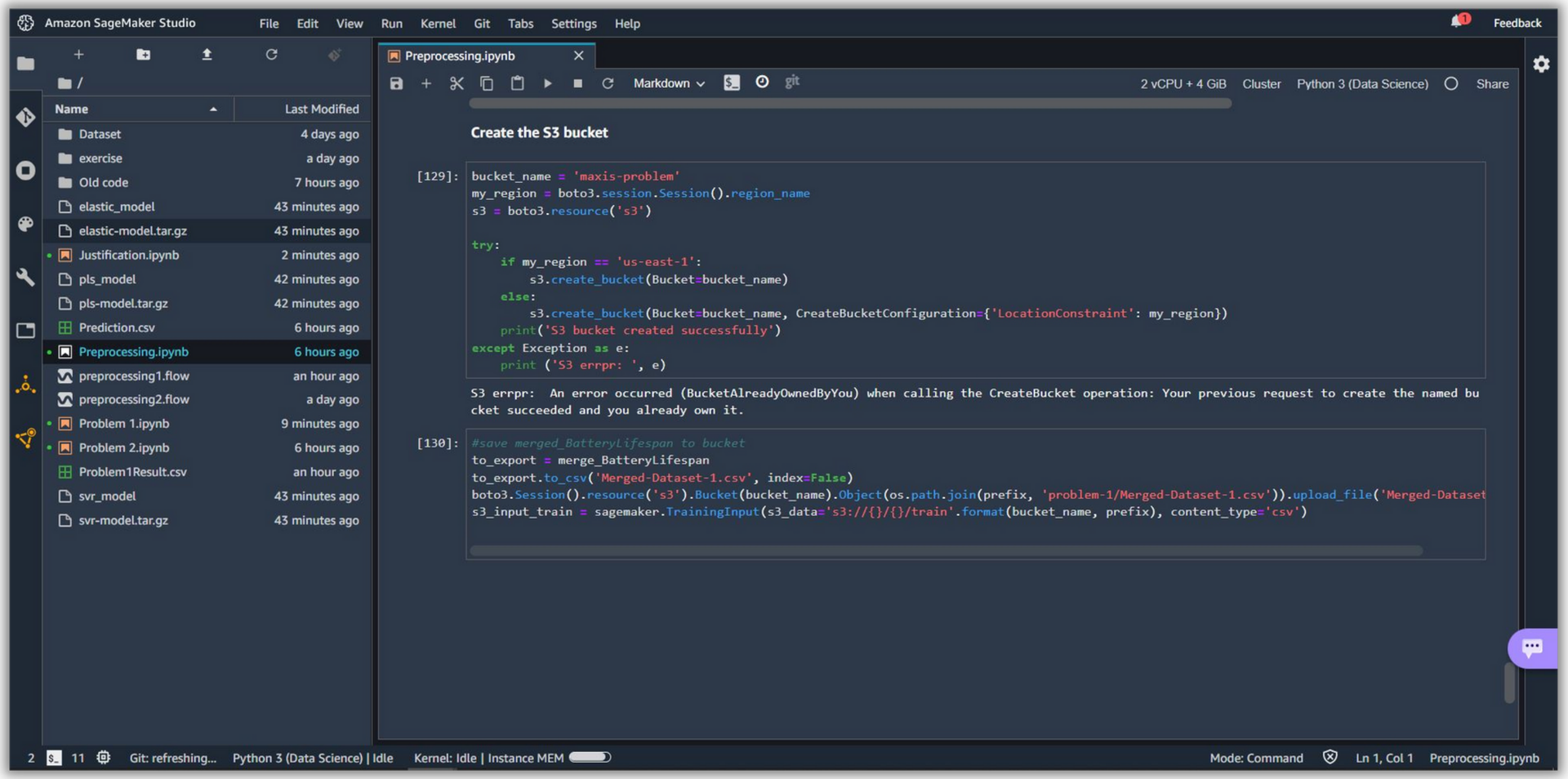
The input features have low VIF score (<10), higher precision of estimated coefficients and statistical power of regression model.

Bias Report



Check the possibility that the model returns an incorrect or non-optimal result because of systematic inflection in processing that produces results that are inconsistent with reality

AWS Sagemaker Studio



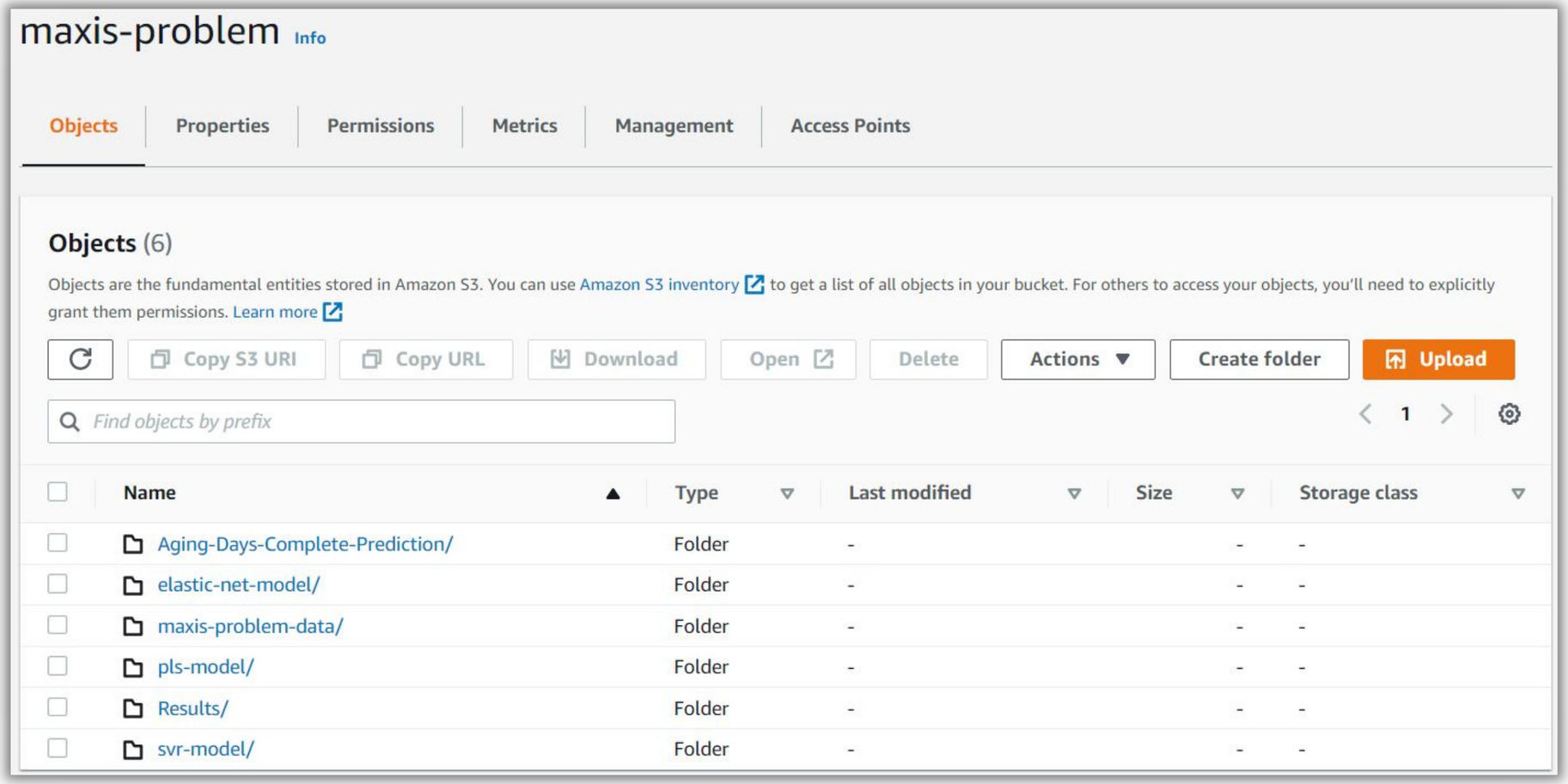
Used to:

Manipulate the datasets

Build and train models

Interact with other AWS Services

AWS S3 Bucket



Used to store:

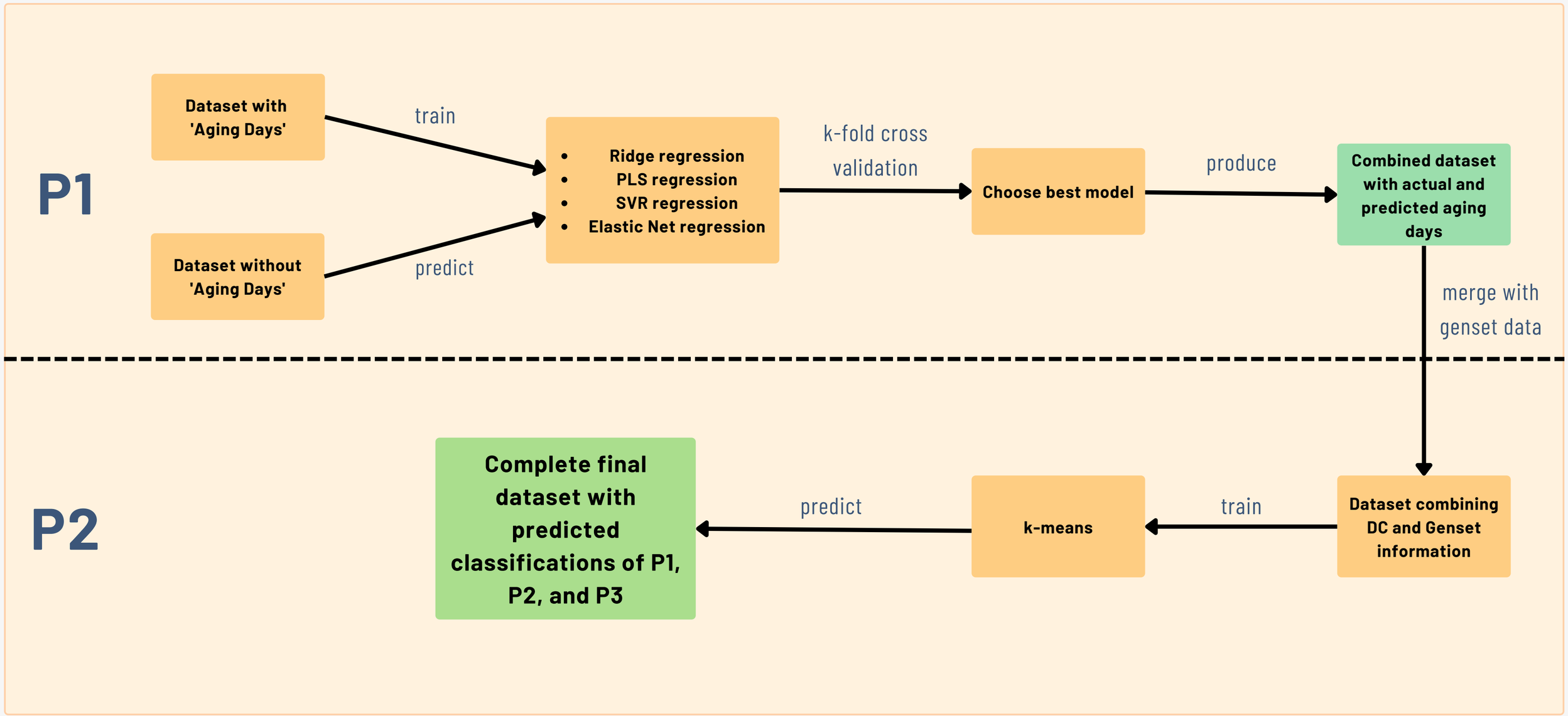
Cleaned Datasets from Data Wrangler

Predicted Values obtained from models

Models

PREDICTING BATTERY AGING

Dataset Lifecycle



DATA MERGING (P1)

01

Cooling System.csv

LRD
Site Type
Cooling System

13317 x 3

02

Battery Health.csv

LRD
Backup Hour
Aging Days
Power Event Count

2774 x 4

03

DC Report.csv

LRD
Admin Status
DC Brand
DC Capacity
Total Current Load
Total Blank
DC Technology

10010 x 7

04

merged_BatteryLifespan

LRD
Backup Hour
Aging Days
Power Event Count
Admin Status
DC Brand
DC Capacity
Total Blank
DC Technology
Site Type
Cooling System

1634 x 11

DATA MERGING (P2)

01 02 03

Predicted Aging Days.csv

LRD
Backup Hours
Aging Days
Power Event Count
Admin Status
Total Bank
DC Technology
DC Brand
Cooling System
Site Type
DC Capacity

5545 x 21

Genset.csv

LRD
Genset Admin Status

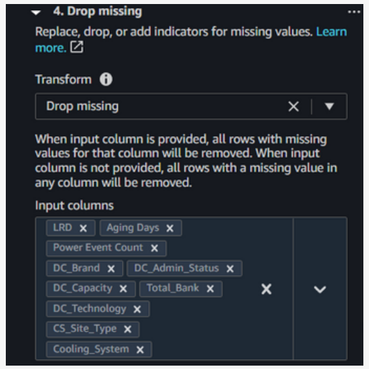
24464 x 2

ONE HOT ENCODING
(Huawei, MSB, Enersys, Narada)
(HEX, FCU, FCUH, AC)
(Vacant land, Rooftop, Inbuilding)
(100.0, 150.0, 155.0)

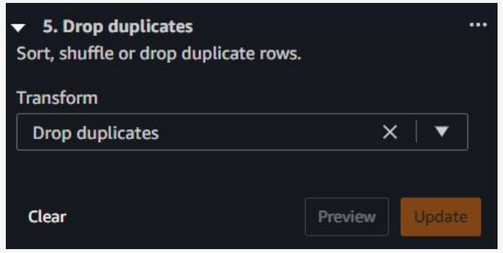
merged_PriorityData
Cluster
LRD
Backup Hours
Aging Days
Power Event Count
Admin Status
Total Bank
DC Technology
DC Brand
Cooling System
Site Type
DC Capacity
Genset Admin Status

5545 x 22

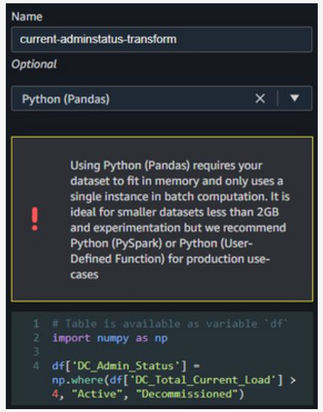
Data Preprocessing



Remove entries with null value in columns



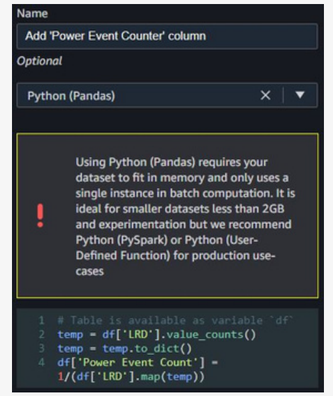
Remove duplicated entries



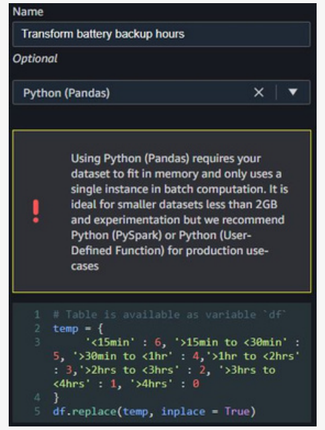
Feature Engineering - Update DC status based on DC_Total_Current Load

DATA CLEANING - OTHER FEATURES

Feature Creation - Create power event count columns based on entries of LRD



Changing categorical data to numeric for Ordinal Features (Backup Hours)



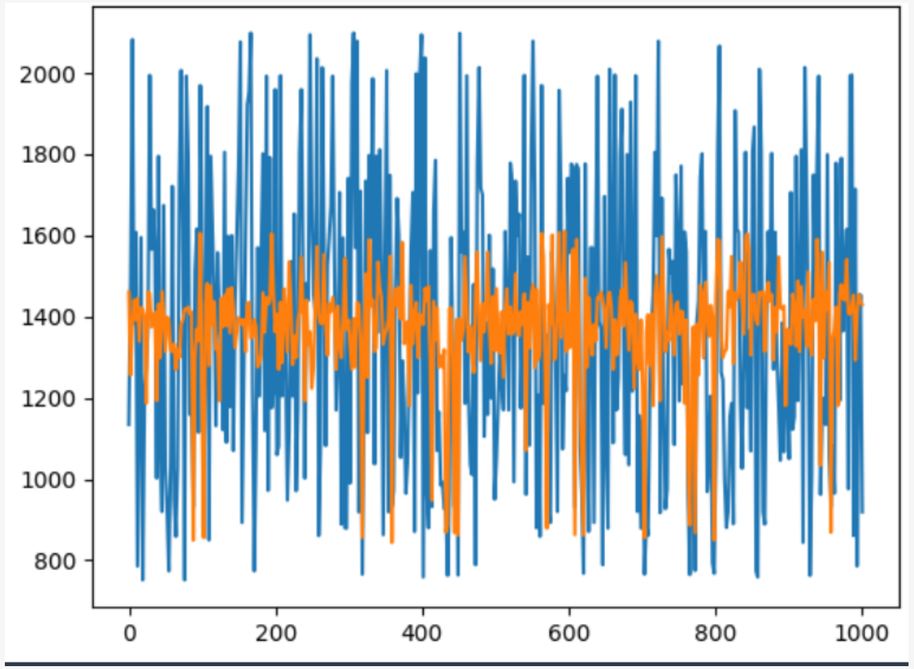
Feature Transformation for Problem 2 (Inversely proportional feature - Aging Days)

```
# KNN_X.drop(['LRD'], axis = 1, inplace = True)
KNN_X['Aging Days'] = KNN_X['Aging Days'].apply(lambda a: 1/a)
```

Removing Outliers

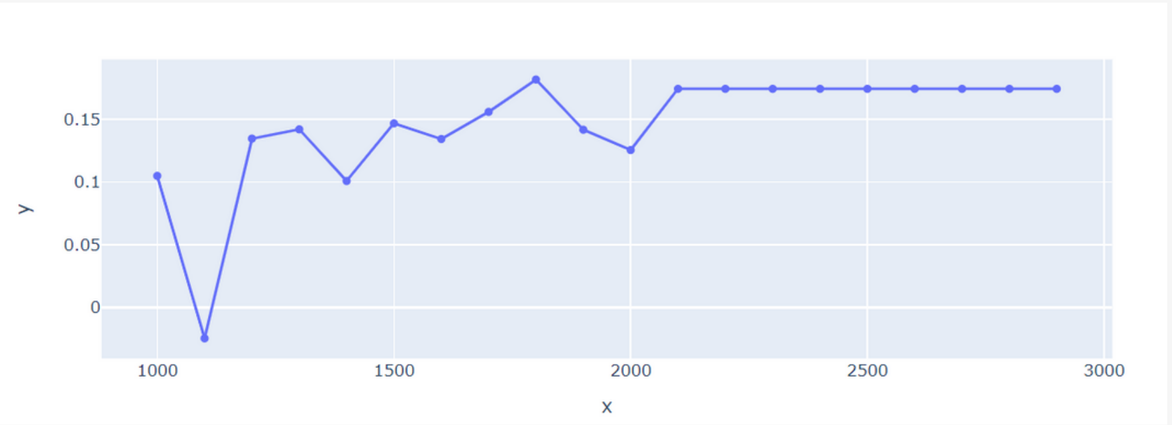
Removal of y-value outlier

Plot of predicted y-values (orange) vs actual value (blue)



Predicted: Compressed

Actual: Spread out



The plot of accuracy vs the upper boundary where the optimal upper boundary is 2100 (Aging Days). So, we will only use entries with aging days between 700 and 2100.

Feature Selection

Removal of columns

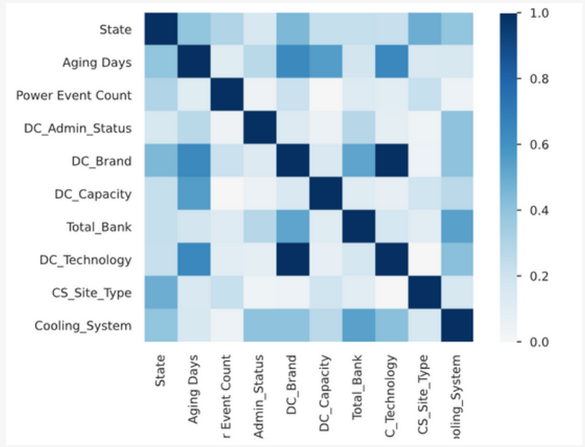
Code for checking accuracy before and after removing "State" columns from battery health check.csv

```
#Initial Accuracy before SMOTE, ONE-HOT, Ensemble Learning ,K-FOLD
#Feature selection
#Removing state

X =  np.array(merge_BatteryLifespan[["State", "Power Event Count", "DC_Admin_Status",
y =  np.array(merge_BatteryLifespan[["Aging Days"]])
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)

LR = LinearRegression()
LR.fit(X_train,y_train)
y_pred =  LR.predict(X_test)
print("R2 score with state feature:", r2_score(y_test, y_pred))

X =  np.array(merge_BatteryLifespan[["Power Event Count", "DC_Admin_Status", "DC_Bra
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.2)
LR.fit(X_train,y_train)
y_pred =  LR.predict(X_test)
print("R2 score without state feature:", r2_score(y_test, y_pred))
```

```
R2 score with state feature: 0.06454440346540657
R2 score without state feature: 0.14734967305514945
```

The accuracy without state feature is significantly higher.

Repeat the same process for each input feature as part of feature selection process as shown by the heat-map correlation



Removal of extreme classes

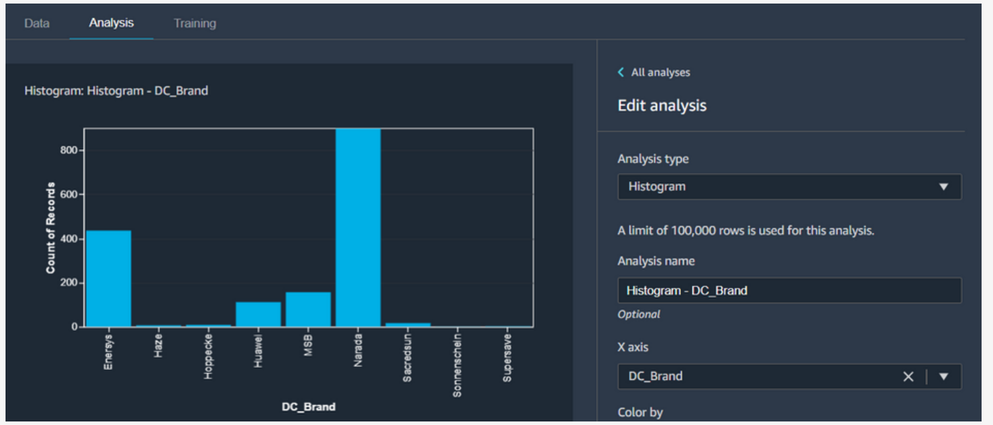
Code to check for distribution of each category in a feature (DC_Brand) and remove extreme cases to improve model performance

```
# Find for the battery brands that are outliers
merge_BatteryLifespan.groupby('DC_Brand')['DC_Brand'].size().sort_values(ascending = True)
```

```
DC_Brand
Haze            1
Sunstone        2
Supersave       2
Sacredsun      11
MSB            53
Huawei        101
Enersys       423
Narada        883
Name: DC_Brand, dtype: int64
```

```
# Remove the battery brand outliers
lst = ['Haze', 'Sunstone', 'Supersave', 'Sacredsun']
merge_BatteryLifespan = merge_BatteryLifespan[~merge_BatteryLifespan['DC_Brand'].isin(lst)]
merge_BatteryLifespan
```

We remove the entries with specific outliers for particular columns. Shown below is the histogram generated using data wrangler



One Hot Encoding and Label Encoding

- **Deal with categorical features**
- **Makes training data more useful and expressive and allows for easy rescaling**
- **Helps improve data accuracy**

One Hot Encoding

Convert each categorical value into a new categorical column and assign a binary value of 1 or 0

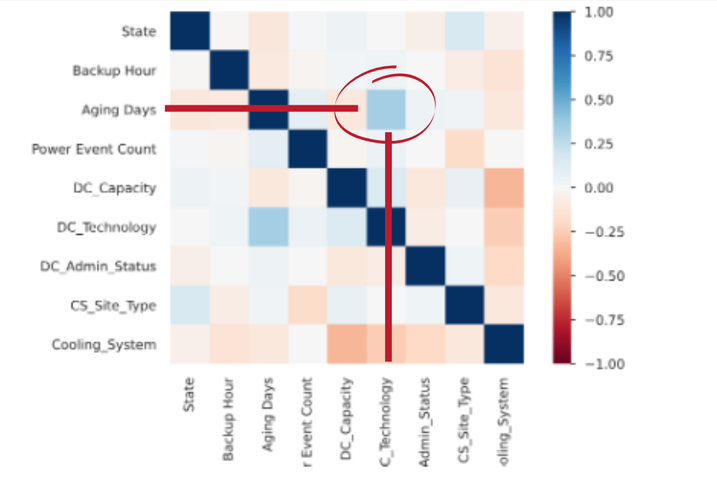
DC Brand
Cooling System
Site Type
DC Capacity

Label Encoding

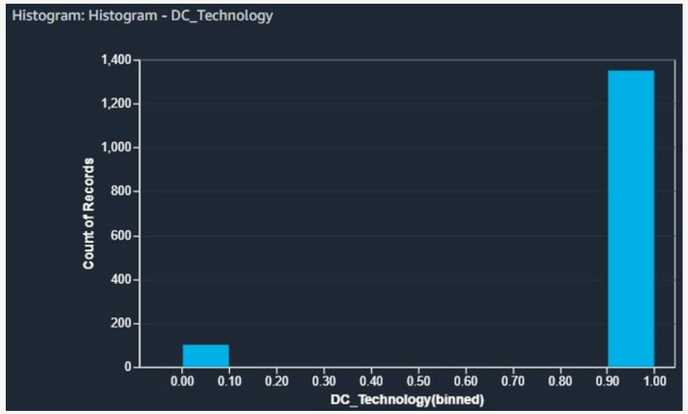
Convert labels to numeric values which is a machine-readable form

Admin Status
DC Technology
Backup Hours

Imbalanced Data - SMOTE (DC Technology)



The feature that is most correlated to the 'Aging Days' is the DC_Technology

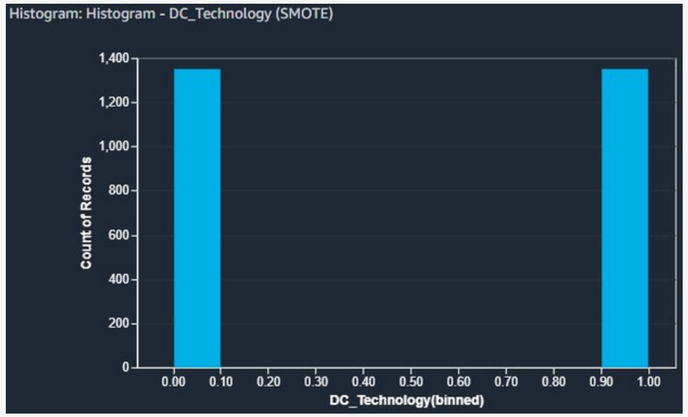


1451 entries:

1350 VLRA

101 Lithium

SMOTE



2700 entries:

1350 VLRA

1350 Lithium

SMOTE focuses on the feature space to generate new instances with the help of interpolation between the positive instances that lie together. This will increase number of entries (row) used in training.

JUSTIFICATION FOR P1 MODEL

RIDGE REGRESSION

Perform L2 regularization to prevent overfitting and multicollinearity

PARTIAL LEAST SQUARE(PLS) REGRESSION

Deal with multicollinearity and several independent variables. Supervised nature of Dimensionality Reduction

SUPPORT VECTOR REGRESSION(SVR)

No correlation between input dimensions and computational complexity. High generalization capability

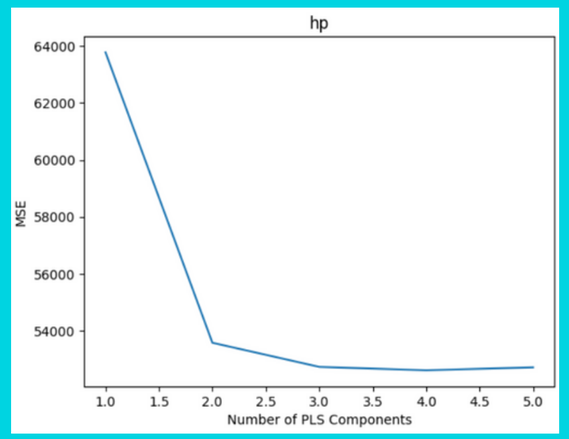
ELASTIC NET REGRESSION

Permits balancing both penalties, which can lead to higher performance than a model with just one penalty on some tasks.

MODEL TUNING

PARTIAL LEAST SQUARE(PLS) REGRESSION (NUMBER OF PLS COMPONENTS)

A graph of MSE against Number of PLS Components is plotted. From the graph below, 4 PLS Components are the best amount for the model.



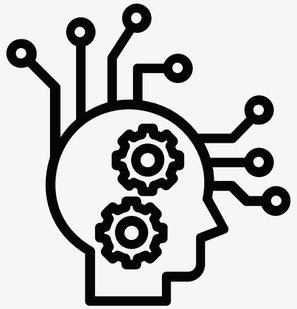
ELASTIC NET REGRESSION (ALPHA)

The higher the alpha, the more your elastic net will have the L1 sparsity feature.

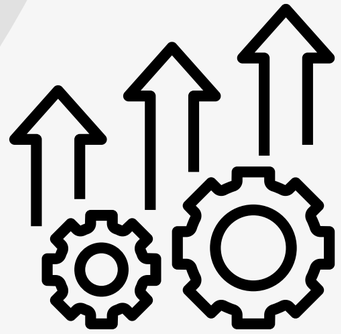
ElasticNetCV is a cross-validation class that can search multiple alpha values and applies the best one.

The alpha value for this model is 0.241.

```
#Elastic Net Regression
from sklearn.linear_model import ElasticNet, ElasticNetCV

enet_cv_model_alpha = ElasticNetCV(cv = 20).fit(X_train, y_train)
enet_model = ElasticNet(alpha = enet_cv_model_alpha.alpha_).fit(X_train, y_train)
```



MODEL IMPROVEMENT



RIDGE REGRESSION (PIPELINING)

Pipeline chain multiple estimators into one.

Pipelines help avoid leaking statistics from your test data into the trained model in cross-validation, the same samples are used to train the transformers and predictors, enhancing the safety of the model.

```
# Ridge Regression
from sklearn.linear_model import Ridge

#Ridge regression using pipeliner
pipeline = make_pipeline(StandardScaler(), Ridge(alpha=1.0))
pipeline.fit(X_train, y_train)
```

SUPPORT VECTOR REGRESSION (TRANSFORM)

Feature Scaling basically helps to normalize the data within a particular range as SVR only accept scale inputs. Scaling help save computational time

```
#Support Vector Regression (SVR)
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVR

sc_X = StandardScaler()
sc_y = StandardScaler()
#Tranform data before feeding model to suit the input
SVR_X = sc_X.fit_transform(X_train)
SVR_y = sc_y.fit_transform(y_train.values.reshape(-1,1))

regressor = SVR(kernel = 'rbf')
regressor.fit(SVR_X, SVR_y.ravel())
```

ERROR METRIC: R2 SCORE

Error metric for regression:

- ✗ Mean Absolute Error (MAE)
- ✓ R Squared Score (R2)
- ✗ Root Mean Squared Error (RMSE)

Explanation:
Compare error metric before and after divide by 365 (Days vs Years) prediction

```
y_test = y_test/365
final_prediction = final_prediction / 365

print("\nError metrics using Aging Years: ")
print("mae: ", mean_absolute_error(y_test, final_prediction))
print("r2: ", r2_score(y_test, final_prediction))
print("rmse: ", math.sqrt(mean_squared_error(y_test, final_prediction)))
```

Justification:
RMSE and MAE scores are based on scale of output. The bigger the scale, the smaller the error due to generalization while r2 is constant

```
Error metrics using Aging Days:
mae:  164.93074456900686
r2:  0.5880075508072624
rmse:  234.91379659289217

Error metrics using Aging Years:
mae:  0.45186505361371754
r2:  0.5880075508072623
rmse:  0.6435994427202525
```

ENSEMBLE LEARNING

Bagging Theory

Learns models independently from each other in parallel and combines them following deterministic averaging process

Hard Code

From scratch implementation of bagging method to better control the flow.

Use 4 models for prediction

Fit each model with training data, then make prediction for X_test

Building dataframe to store all predictions

Combine predictions from earlier into a single dataframe along with Actual Aging Days

Finding mean of 4 predictions

Calculate mean of four predictions into 1 final prediction

Determine accuracy using r2 score

Calculate r2 score of averaging final prediction and actual output

	Ridge	PLS	SVR	Enet	mean	Actual Days
0	1408.0	1367.0	1516.0	1387.0	1420.0	1209.0
1	1235.0	1203.0	1224.0	1199.0	1215.0	1440.0
2	984.0	1014.0	964.0	988.0	988.0	962.0
3	1417.0	1412.0	1699.0	1342.0	1468.0	1481.0

Figure 1.0: Prediction from all Models

```
final accuracy based on ensemble learning:
mae:  159.27084048148149
r2:  0.6518046627152719
rmse:  219.640341652252
```

Figure 2.0: Accuracy of ensemble learning

COMPARING ACCURACY

K-Fold Cross Validation

Abstract

Use K-Fold to compare accuracy of four models: Ridge, PLS, SVR, Elastic Net Regression

```
#Determine best model using K-fold
accumulated_scores = {
'Ridge': np.max(cross_val_score(pipeline, X_test, y_test, scoring = 'r2', cv=cv, n_jobs=-1)),
'PLS' : np.max(cross_val_score(pls, X_test, y_test, scoring = 'r2', cv = cv, n_jobs = -1)),
'SVR' : np.max(cross_val_score(regressor, sc_X.fit_transform(X_test), sc_y.fit_transform(y_test.values.reshape(-1,1))
'ENET' : np.max(cross_val_score(enet_model, X_test, y_test, scoring = 'r2', cv = cv, n_jobs = -1))
}
print(accumulated_scores)
best = max(accumulated_scores, key=accumulated_scores.get)
```

```
{'Ridge': 0.7282896322263149, 'PLS': 0.7452211888375426, 'SVR': 0.8354360761444259, 'ENET': 0.640989636185199}
```

K-Fold Background Theory

Resampling procedure used to evaluate machine learning models on a **limited data sample**

```
# defined K fold for croos validaton
cv = KFold(n_splits = 12, shuffle = True)
```

K=12

k refers to the number of groups that a given data sample is to be split into

Use highest accuracy model for prediction

Use highest accuracy model to make final prediction. Determine using dictionary

```
#input features to predict
X_pred = df_pred.drop(['LRD'], axis = 1)

#To store prediction
y_pred = []
if best == 'Ridge':
  y_pred = pipeline.predict(X_pred)

elif best == 'PLS':
  y_pred = pls.predict(X_pred)

elif best == 'SVR':
  y_pred = sc_y.inverse_transform(regressor.predict(sc_X.fit_transform(X_pred)).reshape(-1,1))

else:
  y_pred = enet_model.predict(X_pred)

print("Best model to use is ", best)
print("Best possible accuracy (R2) of model using k-fold ", accumulated_scores[best])
```

```
Best model to use is  SVR
Best possible accuracy (R2) of model using k-fold  0.8354360761444259
```

MODELS ACCURACY PLOT

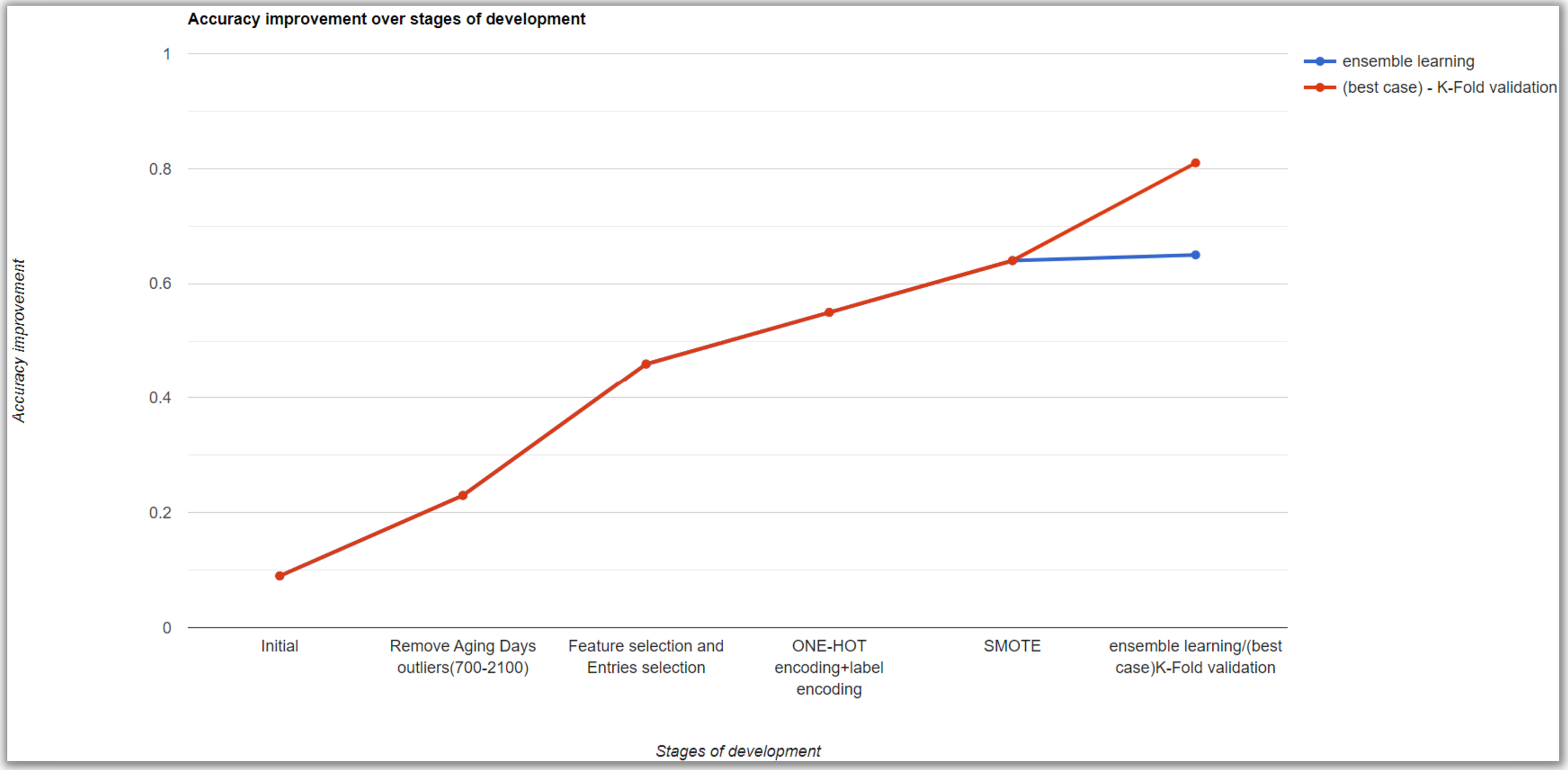
Ensemble learning use average of all model and expecting a higher accuracy. However, in practice, it yields a lower accuracy.

K-Fold Validation yields higher accuracy for each model.

```
Accuracy_plot = pd.DataFrame()
Accuracy_plot = Accuracy_plot.append(accumulated_scores , ignore_index=True)
Accuracy_plot['Ensemble Learning'] = r2
Accuracy_plot = Accuracy_plot.rename(index={0: 'Accuracy'})
Accuracy_plot
```

	Ridge	PLS	SVR	ENET	Ensemble Learning
Accuracy	0.72829	0.745221	0.835436	0.64099	0.651805

Accuracy Improvement Over Time



CATEGORIZING SITES INTO PRIORITIES

Problem 2 Model

K-Means Clustering

K-means clustering is a type of unsupervised learning model which is used when the data is unlabeled. The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable K. In our model, we put our K value as 5. We choose k-means clustering because it can easily adapt to new examples.

P1 (Dangerous)

Genset administrative status is NOT active.

Aging Days >= 1532

P2 (Risky)

Genset administrative status is active.

1063 < Aging Days < 1532

P3 (Safe)

Genset administrative status is active.

Aging Days <= 1063

JUSTIFICATION FOR PROBLEM 2 MODEL

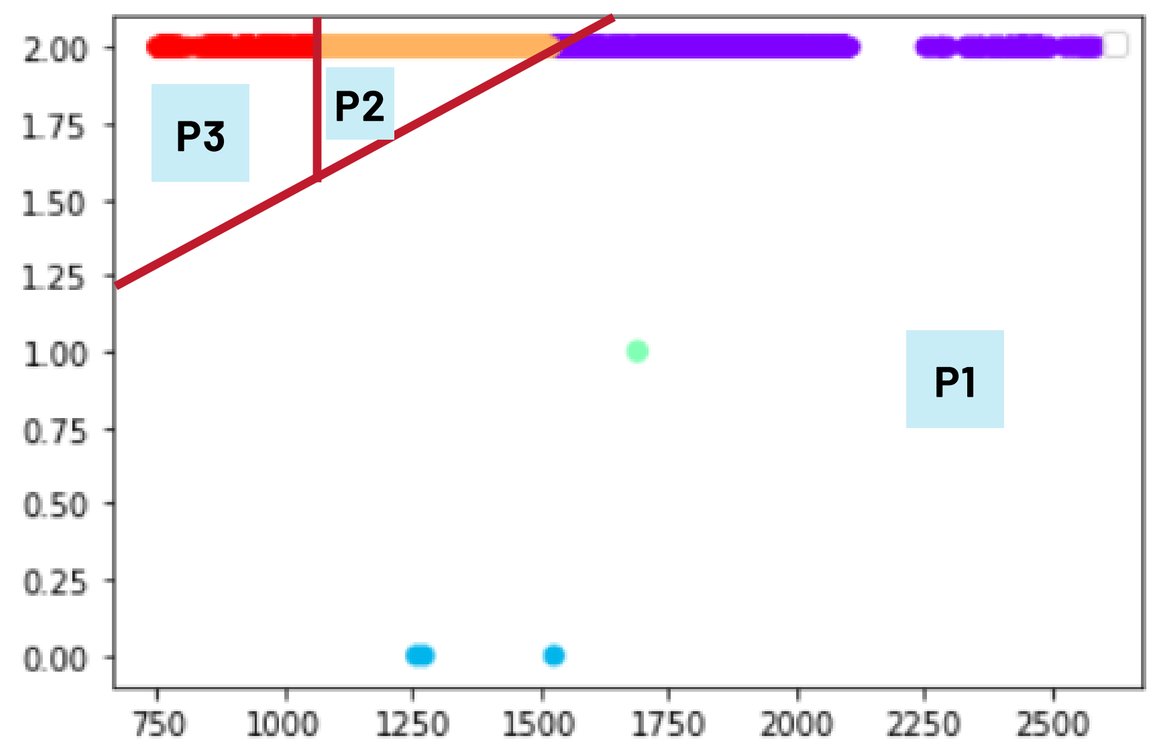
K-means clustering performs effectively compared to linear models when dealing with unlabeled datasets.

K-means clustering has a lower computational cost as it performs well and quickly for large datasets.

Clusters produced using K-means clustering are simple to understand and to depict.

K-means clustering enhances the accuracy of clustering and guarantees the availability of data for a certain problem domain.

Why K = 5?



Visualization of graph

By visualizing the graph, we can see 3 clusters for genset administrative status clearly. The topmost cluster is further divided into 3 clusters, making the total number of clusters to be 5.

Value must be greater than 3

Chosen number must be greater than number of desired classes which is 3

Value preferably should be odd

As K increases, accuracy increases due to majority averaging. Eventually, we begin to witness an increasing number of errors To take a majority vote among labels, we usually make K an odd number to have a tiebreaker.

PROGRAM DEMONSTRATION

BUSINESS VALUES

EXCEPTION HANDLING (WARNING SUPRESSION)

Suppress warnings generated to avoid:

- Warnings from condensing the output
- Avoid errors during model deployment in future

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
  /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692:
    FutureWarning,
  /usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:1692:
    FutureWarning,
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/cross_decomposition/_pls.py:93: RuntimeWarning: invalid value encountered in true_divide
  y_weights = np.dot(Y.T, x_score) / np.dot(x_score.T, x_score)
/usr/local/lib/python3.7/dist-packages/sklearn/cross_decomposition/_pls.py:325: RuntimeWarning: invalid value encountered in true_divide
  x_loadings = np.dot(x_scores, Xk) / np.dot(x_scores, x_scores)
/usr/local/lib/python3.7/dist-packages/sklearn/cross_decomposition/_pls.py:334: RuntimeWarning: invalid value encountered in true_divide
```

Warning encountered when finding optimal number of PLS components

Benefits (Features)

Benefits gained when model is deployed in real world

Data Tolerance

Data cleaning accounts for:

- Outliers
- Extreme classes
- Small dataset
- Features importance
- Feature engineering
- Categorical data encoding
- Imbalance data

Integrated with AWS

Utilized cloud infrastructure:

- Easy scalability
- Pay as used
- Unlimited storage capacity
- Easy deployment
- Back-up and restore
- Data security

Accuracy Reinforcement

Accuracy is reinforced:

- Ensemble Learning
- Model Tuning
- Multiple Model Training
- Accuracy Comparison
- K-Fold Validation
- 0.65 - 0.81

Value Proposition

Predict the aging days of old equipment without installation date captured and new equipment for maintenance

Improve the maintenance scheduling system

Reduce expenditure by reducing unnecessary inspection and maintenance services

Prevent service disruption caused by power supply issues in the next 5 years

Ensure customer satisfaction leading to increased number of subscribers and loyalty

Limitation of Proposed Solution

LIMITATION OF PROPOSED SOLUTION

Unable to predict foreign entries

Since no past data are provided during training, entries whose features are new categories are highly susceptible to error.

Prone to overfitting

Our model is optimized to train upon limited size dataset using SMOTE sampling techniques that will lead to overfitting

Increase memory consumption and computational time

Use of one-hot encoding and SMOTE will generate new entries and columns to improve accuracy at the expense of space and time

Inconsistent Accuracy

Models accuracy varies in ranges of 0.65 - 0.81 instead of showing consistent result due to sparse dataset

THANK YOU