團隊測驗報告

報名序號：111921

團隊名稱：__ 亞柏特 __

註1：請用本PowerPoint 文件撰寫團隊程式說明，**請轉成PDF檔案繳交**。
註2：依據競賽須知第七條，第4項規定：
測試報告之簡報資料不得出現企業、學校系所標誌、提及企業名稱、學校系所、教授姓名及任何可供辨識參賽團隊組織或個人身分的資料或資訊，違者取消參賽資格或由評審會議決議處理方式。

一、資料前處理

紅字為最終使用的方法
藍字為探索過程參考之其他方法

一、探索性分析

- 請詳見 ./Code/EDA.ipynb

二、遺失值補值

- 找到遺失值有存在的row，並透過KNN演算法，取得相近row的特徵進補值
- 直接刪除有遺失值的row
- 差補法

三、異常值偵測

- SUOD : Accelerating Large-scale Unsupervised Heterogeneous Outlier Detection
- ECOD : Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions
- EllipticEnvelope: An object for detecting outliers in a Gaussian distributed dataset.

一、資料前處理

紅字為最終使用的方法
藍字為探索過程參考之其他方法

四、變數轉換:

根據偏態係數將變數轉換成常態分配

- Yeo-Johnson
- Box-cox

五、創建分群特徵

因發現雖然資料為數值變數，但有多特徵都集中在特定數值上，經評估將數值資料轉換成「類別型變數」將有助於模型的訓練，故分別對單一欄位和所有欄位做「分群」產出數個類別型資料

- K-means: 透過自訂的函數，算出最適當的分群數 k，講 kmeans 流程自動化

六、變數標準化

我們發現同時對特徵和目標變數做標準化有助於模型訓練

- Z-score
- MinMax

一、資料前處理

紅字為最終使用的方法
藍字為探索過程參考之其他方法

- **七、變數篩選：**
  - 剔除變異數過小的變數
  - 透過 XGBoost 對 raw data 進行訓練，剔除最差變數，再對剔除該變數後的資料集做訓練，最後剔除最不好的變數，以上不斷循環，直到剩下50個(可自行調整定義)變數。
  - Mutual information
  - Tree model 的 importance weight

二、演算法和模型介紹

- **模型預測目標變數(Y)：共計以下六組**

sensor_point5_i_value
sensor_point6_i_value
sensor_point7_i_value
sensor_point8_i_value
sensor_point9_i_value
sensor_point10_i_value

- **預備模型：經個別訓練後，篩選以下六組模型預測效度較佳，作為後續訓練模型之預備模型**

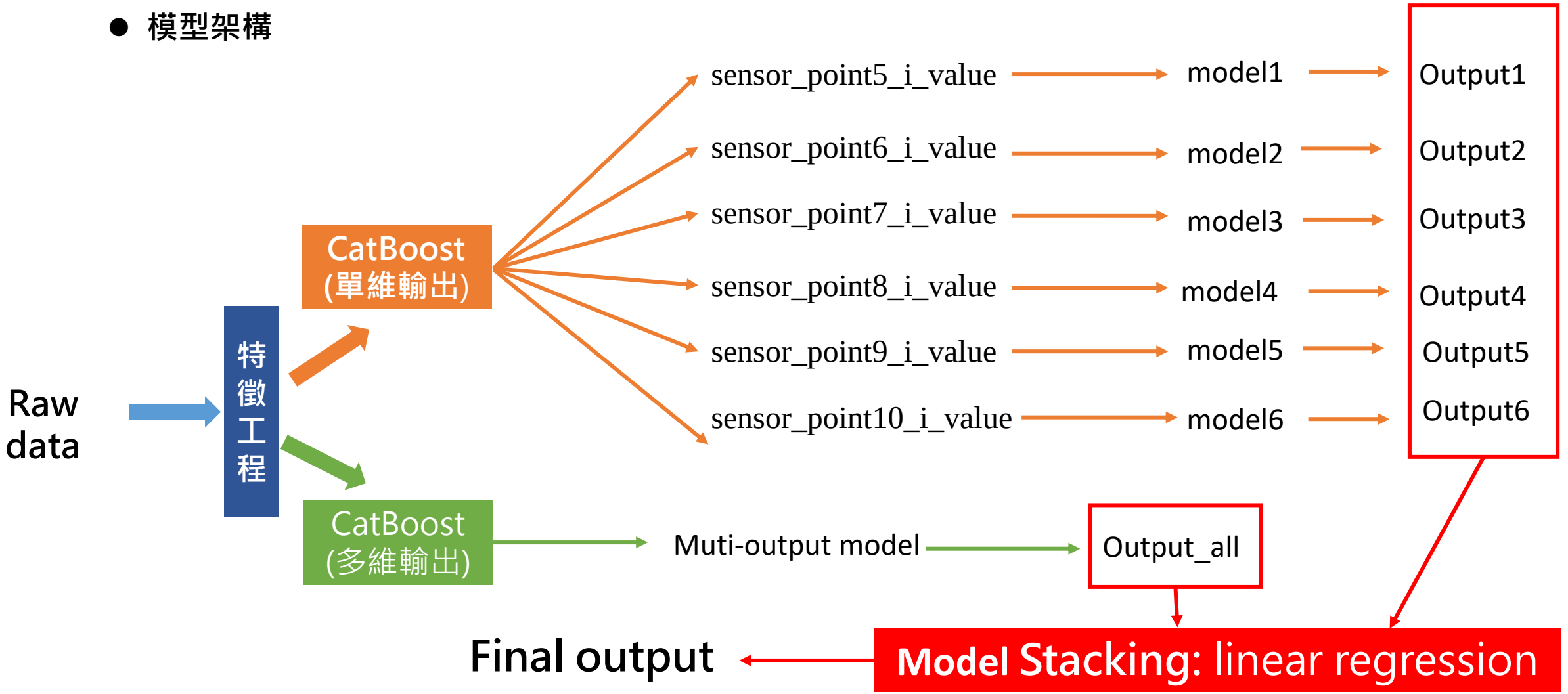
Lasso Regression(L1)
Ridge Regression(L2)
AdaBoostRegressor
KNeighborsRegressor
eXtreme Gradient Boosting(XGBoost)
CatBoostRegressor

二、演算法和模型介紹

- 模型：CatBoost
- 超參數選擇：Random search
- robust: 根據 cross validation 決定最佳模型，避免過擬和

二、演算法和模型介紹

- 模型架構



三、預測結果

- 訓練資料集 RMSE:

```
sensor_point5_i_value      6.285572
sensor_point6_i_value      7.733133
sensor_point7_i_value     12.763048
sensor_point8_i_value     10.315191
sensor_point9_i_value      9.621170
sensor_point10_i_value     9.119610
dtype: float64
```

- 測試資料集
(請參考111921_TestResult)

	A	B	C	D	E	F	G
1	No	sensor_point5_i_value	sensor_point6_i_value	sensor_point7_i_value	sensor_point8_i_value	sor_point9_i_val	sensor_point10_i_value
2	1	52.48210354	64.81554536	70.58318079	39.07227972	66.46720232	52.7213692
3	2	62.08525744	72.99991922	84.18755593	46.90447159	77.16502949	76.86082632
4	3	64.43664168	72.87884391	84.80557682	46.60079117	77.34389171	73.92450494
5	4	79.72294301	55.18569609	77.3355064	42.43928531	62.80858383	66.32231043
6	5	79.72294301	55.18569609	77.3355064	42.43928531	62.80858383	66.32231043
7	6	79.99131869	55.81775232	76.53835387	42.99571137	66.31495987	67.52630328
8	7	78.59014151	57.92364728	76.50558402	42.46220585	64.00889769	69.31571568
9	8	85.23890063	79.56466371	85.88698172	74.03622187	82.59790071	90.00714111
10	9	82.68631402	72.78459735	104.1491622	81.59926026	72.29827155	81.66090864
11	10	82.57582144	73.21660668	105.5772954	77.39153615	79.04923197	81.65780274
12	11	81.59117344	73.64752946	105.6066279	79.25155272	75.44805983	84.32197012
13	12	80.36583926	74.14986934	105.0530194	78.41027535	79.44614646	84.00348439
14	13	81.75285042	76.15422979	103.9260755	78.23480977	83.28700839	84.21397459
15	14	81.82746228	86.13993224	88.5141235	74.53434665	90.82218296	97.33195591
16	15	80.79328201	86.3851298	88.29621933	75.511211	88.30890165	97.18034059
17	16	82.80807411	79.78078904	87.00299879	68.12385183	94.82772244	90.47439025

四、補充說明-演算法和模型介紹

一、透過GridSearchCV方式進行交叉驗證，計算各模型在不同目標變數(Y)下的最佳超參數，考量訓練資料筆數較少，交叉驗證以五折交叉驗證進行訓練，Scoring設定為'neg_root_mean_squared_error'。

```
grid_search_Lasso = GridSearchCV(Lasso(),
                                 param_grid = {'alpha':[0.01,0.1,0.5,1,5,7,10,30,100,500]},
                                 n_jobs = -1,
                                 scoring = 'neg_root_mean_squared_error',
                                 cv = 5
                                 )

grid_search_Ridge = GridSearchCV(Ridge(),
                                 param_grid = {'alpha':[0.01,0.1,0.5,1,5,7,10,30,100,500]},
                                 n_jobs = -1,
                                 scoring = 'neg_root_mean_squared_error',
                                 cv = 5
                                 )


grid_search_XGB = GridSearchCV(xgb.XGBRegressor(),
                                 param_grid = {'learning_rate':np.array([0.2]),
                                               'n_estimators':np.array([30, 100, 200, 500]),
                                               'max_depth':np.array([3, 5, 13, 20]),
                                               'min_child_weight':np.array([3, 10]),
                                               'random_state':[1]
                                               },
                                 n_jobs = -1,
                                 scoring = 'neg_root_mean_squared_error',
                                 cv = 5
                                 )
```

四、補充說明-演算法和模型介紹

一、透過GridSearchCV方式進行交叉驗證，計算各模型在不同目標變數(Y)下的最佳超參數，考量訓練資料筆數較少，交叉驗證以五折交叉驗證進行訓練，Scoring設定為'neg_root_mean_squared_error'。

```
grid_search_KN = GridSearchCV(KNeighborsRegressor(),
                              param_grid = {'n_neighbors':np.array([2, 3, 5, 7]),
                                            'leaf_size':np.array([20, 30, 50]),
                                            'weights':['uniform','distance']
                                           },
                              n_jobs = -1,
                              scoring = 'neg_root_mean_squared_error',
                              cv = 5
                              )

grid_search_Ada = GridSearchCV(AdaBoostRegressor(),
                               param_grid = {'n_estimators':np.array([100, 300, 500]),
                                             'learning_rate':np.array([0.2, 0.5]),
                                             'random_state':[1]
                                             },
                               n_jobs = -1,
                               scoring = 'neg_root_mean_squared_error',
                               cv = 5
                              )

grid_search_Cat = GridSearchCV(CatBoostRegressor(),
                               param_grid = {'iterations': [500 ,1000],
                                             'learning_rate': [0.03],
                                             'depth': [2, 5, 8],
                                             'l2_leaf_reg': [0.5, 3],
                                             'eval_metric':['RMSE'],
                                             'random_state':[1]
                                            },
                               n_jobs = -1,
                               scoring = 'neg_root_mean_squared_error',
                               cv = 5
                              )
```

四、補充說明-演算法和模型介紹

二、以遞迴GridSearchCV方式進行循環交叉驗證，分別取得六個目標變數(Y)在六組模型下的最佳超參數。

```
Lasso_params = {'sensor_point5_i_value': {'alpha': 500},
                'sensor_point6_i_value': {'alpha': 100},
                'sensor_point7_i_value': {'alpha': 500},
                'sensor_point8_i_value': {'alpha': 500},
                'sensor_point9_i_value': {'alpha': 500},
                'sensor_point10_i_value': {'alpha': 500}}

Ridge_params = {'sensor_point5_i_value': {'alpha': 0.1},
                'sensor_point6_i_value': {'alpha': 0.1},
                'sensor_point7_i_value': {'alpha': 0.1},
                'sensor_point8_i_value': {'alpha': 0.1},
                'sensor_point9_i_value': {'alpha': 1},
                'sensor_point10_i_value': {'alpha': 0.5}}

XGB_params = {'sensor_point5_i_value': {'learning_rate': 0.2, 'max_depth': 3, 'min_child_weight': 10, 'n_estimators': 30, 'random_state': 1},
              'sensor_point6_i_value': {'learning_rate': 0.2, 'max_depth': 5, 'min_child_weight': 10, 'n_estimators': 30, 'random_state': 1},
              'sensor_point7_i_value': {'learning_rate': 0.2, 'max_depth': 3, 'min_child_weight': 10, 'n_estimators': 500, 'random_state': 1},
              'sensor_point8_i_value': {'learning_rate': 0.2, 'max_depth': 3, 'min_child_weight': 3, 'n_estimators': 30, 'random_state': 1},
              'sensor_point9_i_value': {'learning_rate': 0.2, 'max_depth': 5, 'min_child_weight': 10, 'n_estimators': 500, 'random_state': 1},
              'sensor_point10_i_value': {'learning_rate': 0.2, 'max_depth': 20, 'min_child_weight': 10, 'n_estimators': 30, 'random_state': 1}}
```

四、補充說明-演算法和模型介紹

二、以遞迴GridSearchCV方式進行循環交叉驗證，分別取得六個目標變數(Y)在六組模型下的最佳超參數。

```
KN_params = {'sensor_point5_i_value': {'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'},
             'sensor_point6_i_value': {'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'},
             'sensor_point7_i_value': {'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'},
             'sensor_point8_i_value': {'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'},
             'sensor_point9_i_value': {'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'},
             'sensor_point10_i_value': {'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'}}

Ada_params = {'sensor_point5_i_value': {'learning_rate': 0.2, 'n_estimators': 100, 'random_state': 1},
              'sensor_point6_i_value': {'learning_rate': 0.2, 'n_estimators': 100, 'random_state': 1},
              'sensor_point7_i_value': {'learning_rate': 0.5, 'n_estimators': 100, 'random_state': 1},
              'sensor_point8_i_value': {'learning_rate': 0.5, 'n_estimators': 300, 'random_state': 1},
              'sensor_point9_i_value': {'learning_rate': 0.2, 'n_estimators': 300, 'random_state': 1},
              'sensor_point10_i_value': {'learning_rate': 0.2, 'n_estimators': 100, 'random_state': 1}}

Cat_params = {'sensor_point5_i_value': {'depth': 8, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 3, 'learning_rate': 0.03, 'random_state': 1},
              'sensor_point6_i_value': {'depth': 2, 'eval_metric': 'RMSE', 'iterations': 1000, 'l2_leaf_reg': 0.5, 'learning_rate': 0.03, 'random_state': 1},
              'sensor_point7_i_value': {'depth': 5, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 3, 'learning_rate': 0.03, 'random_state': 1},
              'sensor_point8_i_value': {'depth': 5, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 3, 'learning_rate': 0.03, 'random_state': 1},
              'sensor_point9_i_value': {'depth': 8, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 0.5, 'learning_rate': 0.03, 'random_state': 1},
              'sensor_point10_i_value': {'depth': 8, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 0.5, 'learning_rate': 0.03, 'random_state': 1}}
```

四、演算法和模型介紹

三、建立pipeline，訓練模型執行步驟如下：

1. 多元回歸(Polynomial)：經測試後，以多元二次多項式預測結果最佳，故設定degree = 2。
2. 標準化：將X變數進行標準化，提高模型訓練效率(收斂速度)及變數精準度。
3. 訓練模型將前述步驟四計算得出之超參數放入模型後進行訓練。

(語法範例如下，以XGBoost為例)

```
poly_XGB_reg = Pipeline([('poly',PolynomialFeatures(degree = 2)),
                         ('std_scaler', StandardScaler()),
                         ('xgb_reg', xgb.XGBRegressor(**config.grid_search_XGB.best_params_))
                         ])
```

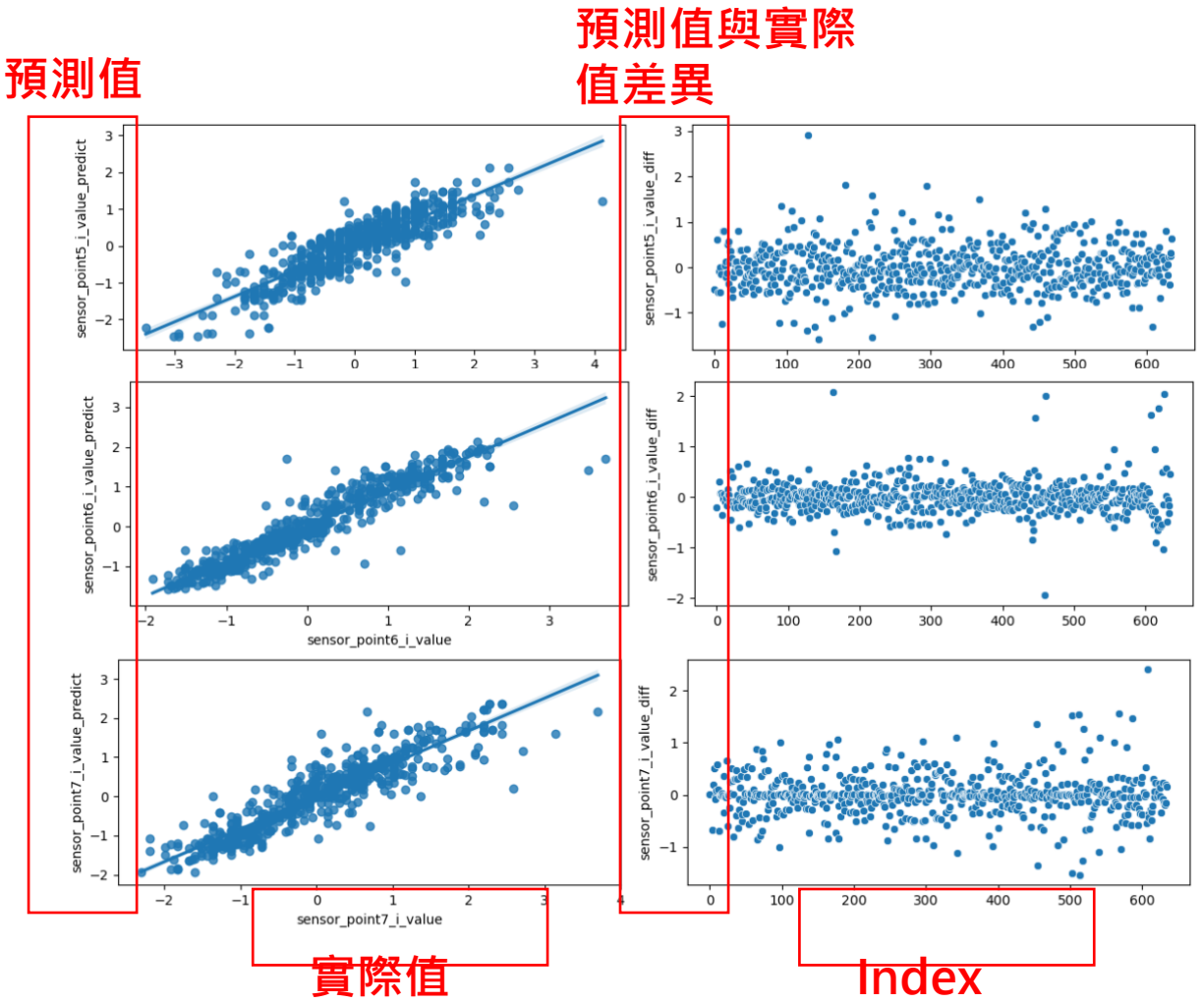
四、補充說明-演算法和模型介紹

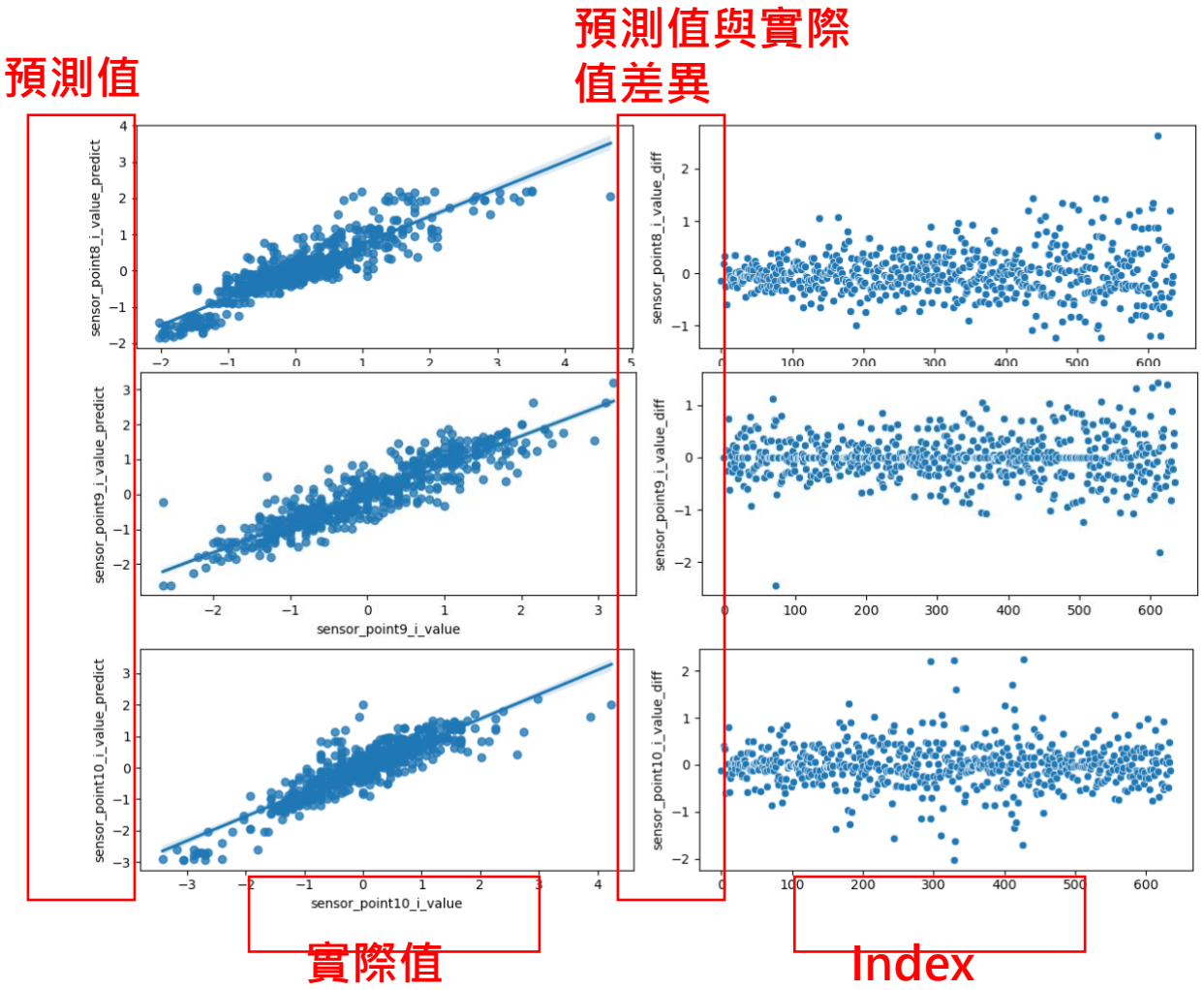
四、透過遞迴方式，以五折cross_val_score(k-folder交叉驗證)計算六個目標變數(Y)的在六組模型下的平均RMSE，計算後之RMSE分布如下，透過下表可透過個別模型的RMSE初步評估各目標變數(Y)的最佳預測模型主要為Lasso、KN、Cat、Ada四組模型。

	Lasso_RMSE	Ridge_RMSE	XGB_RMSE	KN_RMSE	Ada_RMSE	Cat_RMSE
sensor_point5_i_value	1.04	1.08	1.11	1.04	1.11	1.05
sensor_point6_i_value	1.01	1.14	1.02	0.97	1.00	0.93
sensor_point7_i_value	0.94	1.05	1.09	0.96	1.09	1.01
sensor_point8_i_value	0.98	1.16	1.02	0.91	0.96	1.03
sensor_point9_i_value	1.07	1.14	1.17	1.04	1.03	1.05
sensor_point10_i_value	1.02	1.16	1.06	1.06	1.00	1.00

四、補充說明-演算法和模型介紹(介紹方法細節)

五、以預測結果分布狀況，輔以前述步驟六計算之RMSE，挑選各預測目標(Y)下的最佳模型。





四、補充說明-演算法和模型介紹

六、經前述步驟挑選各預測目標(Y)下的最佳模型如下表

預測目標(Y)	最佳模型	超參數
sensor_point5_i_value	CatBoostRegressor	{'depth': 8, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 3, 'learning_rate': 0.03, 'random_state': 1}
sensor_point6_i_value	CatBoostRegressor	{'depth': 2, 'eval_metric': 'RMSE', 'iterations': 1000, 'l2_leaf_reg': 0.5, 'learning_rate': 0.03, 'random_state': 1}
sensor_point7_i_value	KNeighborsRegressor	{'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'}
sensor_point8_i_value	KNeighborsRegressor	{'leaf_size': 20, 'n_neighbors': 7, 'weights': 'distance'}
sensor_point9_i_value	AdaBoostRegressor	{'learning_rate': 0.2, 'n_estimators': 300, 'random_state': 1}
sensor_point10_i_value	CatBoostRegressor	{'depth': 8, 'eval_metric': 'RMSE', 'iterations': 500, 'l2_leaf_reg': 0.5, 'learning_rate': 0.03, 'random_state': 1}}

四、補充說明-演算法和模型介紹

七、考量預測目標(Y)之間可能具有一定程度之相依性，採依序訓練的方式(RegressorChain)，先以模型預測第一個預測目標(Y)，再將第一個預測目標(Y)的預測結果放入X變數中，進行第二個模型預測，以此類推，逐一預測，再將預測結果作為新變數供下個模型進行預測。並採用遞迴方式，選取合計RMSE最低的預測順序。

預測目標(Y)	預測順序
sensor_point5_i_value	6
sensor_point6_i_value	4
sensor_point7_i_value	3
sensor_point8_i_value	2
sensor_point9_i_value	5
sensor_point10_i_value	1

```
# 迴圈依據套用模型計算預測值(Y)，並將預測結果放入X中做為新變數供下一輪模型預測時使用。(最佳模型順序[5, 3, 2, 1, 4, 0])

X_PT = data_x.copy()
predict_y_train = pd.DataFrame({})

for k in range(6):

    with open('../model/model_Y'+ str(config.order_list[k]) + '(Albert).pickle', 'rb') as f:
        model = pickle.load(f)

    # 儲存預測結果
    predict_y_train[config.data_y_col[config.order_list[k]]] = model.predict(X_PT)
    # 把取得的預測值當作變數放進X
    X_PT[config.data_y_col[config.order_list[k]]] = model.predict(X_PT)

    # 清空 model
    del model

# 調整欄位順序
predict_y_train = predict_y_train[config.data_y_col]
display(predict_y_train)
```