กลุ่ม Diabetes Prediction

1. 6610402205 นายรักษิต รุ่งรัตนไชย หมู่ 1
2. 6610402132 นายบวรรัตน์ ตั้งนราธัชกิจ หมู่ 1
3. 6610401985 นายไชยวัตน์ หนูวัฒนา หมู่ 1

วัตถุประสงค์ของระบบต้นแบบ

กลุ่มของพวกเราเล็งเห็นถึงความสำคัญทางด้านสุขภาพของประชาชนคนไทย ในปัจจุบันคนไทยประสบกับปัญหาสุขภาพมากมายแต่หนึ่งปัญหาที่สร้างความเสียหายให้กับสุขภาพ และคนไทยเป็นกันมากที่สุดนั่นคือโรคเบาหวาน พฤติกรรมการใช้ชีวิตของคนไทยในสมัยนี้ก็เป็นปัจจัยหนึ่งที่ทำให้ปริมาณผู้ป่วยโรคเบาหวานเพิ่มมากยิ่งขึ้น พวกเราจึงอยากทำระบบต้นแบบที่ช่วยชี้แนะหรือแนะนำผู้ที่มาใช้งานระบบต้นแบบของเราว่ามีความเสี่ยงที่จะเป็นโรคเบาหวานหรือไม่เพื่อให้เกิดการตระหนักและระมัดระวังในพฤติกรรมการใช้ชีวิตของตนเองซึ่งระบบต้นแบบของเราไม่ได้แม่นยำ 100 เปอร์เซ็นต์ ไม่สามารถใช้วินิจฉัยโรคได้แต่ใช้ประเมินความเสี่ยงเบื้องต้นของผู้ที่มาใช้งานระบบได้หากมีความเสี่ยงก็สามารถให้แพทย์วินิจฉัยอย่างถูกต้องและแม่นยำต่อไป

ลิงค์ไปยังข้อมูลที่จะใช้ในระบบต้นแบบ

Link to data: https://www.kaggle.com/datasets/alexteboul/diabetes-health-indicators-dataset

In [ ]:
```python
import pandas as pd

diabetes = pd.read_csv("diabetes_binary_health_indicators_BRFSS2015.csv")
```

Data Engineer

ตรวจสอบข้อมูลสูญหาย (Missing Value)

In [ ]:
```python
diabetes.isna().sum()
```

Out[ ]:
```
Diabetes_binary         0
HighBP                  0
HighChol                0
CholCheck               0
BMI                     0
Smoker                  0
Stroke                  0
HeartDiseaseorAttack    0
PhysActivity            0
Fruits                  0
Veggies                 0
HvyAlcoholConsump       0
AnyHealthcare           0
NoDocbcCost             0
GenHlth                 0
MentHlth                0
PhysHlth                0
DiffWalk                0
Sex                     0
Age                     0
Education               0
Income                  0
dtype: int64
```

ตรวจสอบข้อมูลซ้อนซ้อน

In [ ]:
```python
diabetes.duplicated().sum()
```

Out[ ]:
```
24206
```

In [ ]:
```python
diabetes.drop_duplicates(inplace=True)
```

อธิบายเพิ่ม: ในครั้งก่อนๆฟีเจอร์ทั้งหมดที่นำมาใช้นั้นเป็นฟีเจอร์ 5 ฟีเจอร์ที่หามาจากการทำ feature importance ที่มีค่าความสำคัญมากที่สุดเรียงลงมา ได้แก่ `GenHlth`, `BMI`, `HeartDiseaseorAttack`, `DiffWalk`, `HvyAlcoholConsump` ซึ่งเมื่อนำไปให้โมเดลเรียนรู้ ผลลัพธ์ที่ได้คือโมเดลมีความแม่นยำไม่มากพอที่จะนำไปใช้งานจริง ในครั้งนี้จึงเปลี่ยนมาใช้วิธีการดูความสัมพันธ์ของแต่ละฟีเจอร์ เพื่อดูว่าฟีเจอร์ใดบ้างที่มีความสัมพันธ์ต่อฟีเจอร์อื่นๆ โดยได้ฟีเจอร์ที่มีความสัมพันธ์กับฟีเจอร์อื่นๆมากที่สุดลงมา ได้แก่ `GenHlth`, `HighBP`, `HighChol`, `BMI`, `Age`, `DiffWalk` และเมื่อนำฟีเจอร์เหล่านี้ไปให้โมเดลเรียนรู้ผลปรากฏว่าโมเดลมีความแม่นยำเพิ่มขึ้นในระดับนึง

In [ ]:
```python
diabetes.corr().sort_values(by=["Diabetes_binary"], ascending=False)
```

Out[ ]:

	Diabetes_binary	HighBP	HighChol	CholCheck	BMI	Smoker	Stroke	HeartDiseaseorAttack	PhysActivity	Fruits	...	AnyHealthcare	NoDocbcCost	GenHlth	MentHlth	PhysHlth	DiffWalk	Sex	Age	Education	Income
Diabetes_binary	1.000000	0.254318	0.194944	0.072523	0.205086	0.045504	0.099193	0.168213	-0.100404	-0.024805	...	0.025331	0.020048	0.276940	0.054153	0.156211	0.205302	0.032724	0.177263	-0.102686	-0.140659
GenHlth	0.276940	0.272784	0.188139	0.062939	0.208351	0.134894	0.169809	0.246328	-0.237511	-0.071221	...	-0.023286	0.149690	1.000000	0.283822	0.516476	0.446696	-0.010519	0.145769	-0.244752	-0.331782
HighBP	0.254318	1.000000	0.283963	0.111259	0.194218	0.074264	0.124558	0.201443	-0.104382	-0.019467	...	0.052044	0.002292	0.272784	0.037482	0.144656	0.211759	0.047119	0.339802	-0.112887	-0.140030
DiffWalk	0.205302	0.211759	0.136045	0.049107	0.182556	0.108144	0.169339	0.202657	-0.235719	-0.029932	...	0.017714	0.106225	0.446696	0.218733	0.466852	1.000000	-0.073405	0.205754	-0.169350	-0.299064
BMI	0.205086	0.194218	0.089734	0.042487	1.000000	-0.009294	0.011006	0.039820	-0.127780	-0.067424	...	-0.008519	0.045795	0.208351	0.068569	0.102768	0.182556	0.030902	-0.049223	-0.074433	-0.069097
HighChol	0.194944	0.283963	1.000000	0.094772	0.089734	0.074583	0.089375	0.176446	-0.063443	-0.026257	...	0.052363	0.003020	0.188139	0.050346	0.111008	0.136045	0.022859	0.263841	-0.050045	-0.062089
Age	0.177263	0.339802	0.263841	0.095942	-0.049223	0.107695	0.128209	0.223912	-0.087881	0.073515	...	0.147465	-0.130115	0.145769	-0.103355	0.095483	0.205754	-0.031862	1.000000	-0.092747	-0.116361
HeartDiseaseorAttack	0.168213	0.201443	0.176446	0.050086	0.039820	0.105169	0.198814	1.000000	-0.073094	-0.006946	...	0.025987	0.021971	0.246328	0.052601	0.170335	0.202657	0.089828	0.223912	-0.082288	-0.122728
PhysHlth	0.156211	0.144656	0.111008	0.040758	0.102768	0.100447	0.140806	0.170335	-0.199307	-0.024441	...	0.002924	0.136421	0.516476	0.340191	1.000000	0.466852	-0.044433	0.095483	-0.127687	-0.240929
Stroke	0.099193	0.124558	0.089375	0.027955	0.011006	0.054414	1.000000	0.198814	-0.059306	-0.004486	...	0.013627	0.028613	0.169809	0.061996	0.140806	0.169339	0.003626	0.128209	-0.064178	-0.117108
CholCheck	0.072523	0.111259	0.094772	1.000000	0.042487	-0.003721	0.027955	0.050086	-0.004555	0.017860	...	0.115498	-0.054128	0.062939	-0.001430	0.040758	0.049107	-0.024332	0.095942	-0.009935	0.001989
MentHlth	0.054153	0.037482	0.050346	-0.001430	0.068569	0.077641	0.061996	0.052601	-0.105914	-0.052191	...	-0.043582	0.181769	0.283822	1.000000	0.340191	0.218733	-0.083836	-0.103355	-0.076122	-0.185689
Smoker	0.045504	0.074264	0.074583	-0.003721	-0.009294	1.000000	0.054414	0.105169	-0.066869	-0.061731	...	-0.013963	0.037335	0.134894	0.077641	0.100447	0.108144	0.096650	0.107695	-0.135657	-0.095314
Sex	0.032724	0.047119	0.022859	-0.024332	0.030902	0.096650	0.003626	0.089828	0.033516	-0.088768	...	-0.021221	-0.046507	-0.010519	-0.083836	-0.044433	-0.073405	1.000000	-0.031862	0.015956	0.130997
AnyHealthcare	0.025331	0.052044	0.052363	0.115498	-0.008519	-0.013963	0.013627	0.025987	0.023959	0.022659	...	1.000000	-0.227469	-0.023286	-0.043582	0.002924	0.017714	-0.021221	0.147465	0.111367	0.146144
NoDocbcCost	0.020048	0.002292	0.003020	-0.054128	0.045795	0.037335	0.028613	0.021971	-0.046440	-0.032387	...	-0.227469	1.000000	0.149690	0.181769	0.136421	0.106225	-0.046507	-0.130115	-0.083260	-0.187577
Fruits	-0.024805	-0.019467	-0.026257	0.017860	-0.067424	-0.061731	-0.004486	-0.006946	0.125023	1.000000	...	0.022659	-0.032387	-0.071221	-0.052191	-0.024441	-0.029932	-0.088768	0.073515	0.084857	0.050907
Veggies	-0.041734	-0.042994	-0.027399	-0.000653	-0.044054	-0.013744	-0.033029	-0.027180	0.135240	0.242941	...	0.020530	-0.019876	-0.094115	-0.042215	-0.045130	-0.063189	-0.066113	-0.003856	0.131624	0.125068
HvyAlcoholConsump	-0.065950	-0.014178	-0.019057	-0.020975	-0.058420	0.096048	-0.021347	-0.035561	0.023378	-0.028221	...	-0.006202	-0.001272	-0.055783	0.016852	-0.036860	-0.047655	0.009435	-0.041018	0.039132	0.071863
PhysActivity	-0.100404	-0.104382	-0.063443	-0.004555	-0.127780	-0.066869	-0.059306	-0.073094	1.000000	0.125023	...	0.023959	-0.046440	-0.237511	-0.105914	-0.199307	-0.235719	0.033516	-0.087881	0.170931	0.165869
Education	-0.102686	-0.112887	-0.050045	-0.009935	-0.074433	-0.135657	-0.064178	-0.082288	0.170931	0.084857	...	0.111367	-0.083260	-0.244752	-0.076122	-0.127687	-0.169350	0.015956	-0.092747	1.000000	0.419045
Income	-0.140659	-0.140030	-0.062089	0.001989	-0.069097	-0.095314	-0.117108	-0.122728	0.165869	0.050907	...	0.146144	-0.187577	-0.331782	-0.185689	-0.240929	-0.299064	0.130997	-0.116361	0.419045	1.000000

22 rows × 22 columns

In [ ]:
```python
# features = ["GenHlth", "BMI", "HeartDiseaseorAttack", "DiffWalk", "HvyAlcoholConsump"] # Feature Importance
features = ["GenHlth", "HighBP", "HighChol", "BMI", "Age", "DiffWalk"] # Correlation
# features = diabetes.drop("Diabetes_binary", axis=1).columns # All
label = "Diabetes_binary"
```

ตรวจสอบและจัดการกับ Imbalance ด้วยการทำ Undersampling

In [ ]:
```python
from imblearn.under_sampling import RandomUnderSampler
resampler = RandomUnderSampler(random_state=1234)

diabetes[features], diabetes[label] = resampler.fit_resample(diabetes[features], diabetes[label])
diabetes.dropna(inplace=True)

diabetes[features].head()
```

Out[ ]:

	GenHlth	HighBP	HighChol	BMI	Age	DiffWalk
8	5.0	1.0	1.0	30.0	9.0	1.0
10	3.0	0.0	0.0	25.0	13.0	0.0
13	4.0	1.0	1.0	28.0	11.0	1.0
15	2.0	1.0	0.0	33.0	6.0	0.0
16	3.0	1.0	1.0	21.0	10.0	0.0

In [ ]:
```python
diabetes.Diabetes_binary.value_counts()
```

Out[ ]:
```
Diabetes_binary
1.0    35097
0.0    35097
Name: count, dtype: int64
```

Traning Data

แบ่งชุดข้อมูลเรียนรู้(Train), ชุดข้อมูลทดสอบ(Test), และชุดข้อมูลตรวจสอบ (Validation)

In [ ]:
```python
from sklearn.model_selection import train_test_split

train, test = train_test_split(diabetes, test_size = 0.3, stratify = diabetes.Diabetes_binary, random_state = 1234)
test, validation = train_test_split(test, test_size = 0.3, stratify = test.Diabetes_binary, random_state = 1234)
```

In [ ]:
```python
len(train), len(test), len(validation)
```

Out[ ]:
```
(49135, 14741, 6318)
```

ตรวจสอบความสมดุลของชุดข้อมูลเรียนรู้

In [ ]:
```python
train.Diabetes_binary.value_counts()
```

Out[ ]:
```
Diabetes_binary
0.0    24568
1.0    24567
Name: count, dtype: int64
```

Model Development

สร้าง Pipeline ของโมเดล Logistic Regression และทำ Feature Engineer ใน Pipeline

In [ ]:
```python
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from sklearn.preprocessing import KBinsDiscretizer
from sklearn.linear_model import LogisticRegression

def create_pipline_logistic():
    return make_pipeline(
        make_column_transformer(
            (KBinsDiscretizer(encode="ordinal", strategy="quantile"), ["BMI"]),
            remainder="passthrough"
        ),
        LogisticRegression(max_iter=500)
    )

logistic_pipeline = create_pipline_logistic()
logistic_pipeline
```

Out[ ]:
```
Pipeline
  columntransformer: ColumnTransformer
    kbinsdiscretizer        remainder
    KBinsDiscretizer        passthrough
  LogisticRegression
```

In [ ]:
```python
%time logistic_pipeline.fit(train[features], train[label])

logistic_train_score = logistic_pipeline.score(train[features], train[label])
logistic_validation_score = logistic_pipeline.score(validation[features], validation[label])
print(f"Train Score: %.3f" % logistic_train_score)
print(f"Validation Score: %.3f" % logistic_validation_score)
```

```
CPU times: total: 31.2 ms
Wall time: 77.3 ms
Train Score: 0.730
Validation Score: 0.728
```

Hyperparameter Tuning

In [ ]:
```python
logistic_pipeline.get_params()
```

Out[ ]:
```
{'memory': None,
 'steps': [('columntransformer',
   ColumnTransformer(remainder='passthrough',
                     transformers=[('kbinsdiscretizer',
                                    KBinsDiscretizer(encode='ordinal'), ['BMI'])])),
  ('logisticregression', LogisticRegression(max_iter=500))],
 'verbose': False,
 'columntransformer': ColumnTransformer(remainder='passthrough',
                   transformers=[('kbinsdiscretizer',
                                  KBinsDiscretizer(encode='ordinal'), ['BMI'])]),
 'logisticregression': LogisticRegression(max_iter=500),
 'columntransformer__force_int_remainder_cols': True,
 'columntransformer__n_jobs': None,
 'columntransformer__remainder': 'passthrough',
 'columntransformer__sparse_threshold': 0.3,
 'columntransformer__transformer_weights': None,
 'columntransformer__transformers': [('kbinsdiscretizer',
   KBinsDiscretizer(encode='ordinal'),
   ['BMI'])],
 'columntransformer__verbose': False,
 'columntransformer__verbose_feature_names_out': True,
 'columntransformer__kbinsdiscretizer': KBinsDiscretizer(encode='ordinal'),
 'columntransformer__kbinsdiscretizer__dtype': None,
 'columntransformer__kbinsdiscretizer__encode': 'ordinal',
 'columntransformer__kbinsdiscretizer__n_bins': 5,
 'columntransformer__kbinsdiscretizer__random_state': None,
 'columntransformer__kbinsdiscretizer__strategy': 'quantile',
 'columntransformer__kbinsdiscretizer__subsample': 200000,
 'logisticregression__C': 1.0,
 'logisticregression__class_weight': None,
 'logisticregression__dual': False,
 'logisticregression__fit_intercept': True,
 'logisticregression__intercept_scaling': 1,
 'logisticregression__l1_ratio': None,
 'logisticregression__max_iter': 500,
 'logisticregression__multi_class': 'deprecated',
 'logisticregression__n_jobs': None,
 'logisticregression__penalty': 'l2',
 'logisticregression__random_state': None,
 'logisticregression__solver': 'lbfgs',
 'logisticregression__tol': 0.0001,
 'logisticregression__verbose': 0,
 'logisticregression__warm_start': False}
```

In [ ]:
```python
from sklearn.model_selection import RandomizedSearchCV

import warnings
warnings.filterwarnings("ignore")
pipeline = create_pipline_logistic()

param_dist = {
     'columntransformer__kbinsdiscretizer__strategy': ['uniform', 'quantile', 'kmeans'],
     'columntransformer__kbinsdiscretizer__encode': ['ordinal', 'onehot', 'onehot-dense'],
     'columntransformer__kbinsdiscretizer__n_bins': [5, 10, 20],
    'logisticregression__C': [0.001, 0.01, 0.1, 1, 10, 100],
    'logisticregression__penalty': ['none', 'l1', 'l2', 'elasticnet'],
    'logisticregression__solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],
    'logisticregression__max_iter': [100, 200, 300],
    'logisticregression__tol': [1e-4, 1e-3, 1e-2]
}

search = RandomizedSearchCV(pipeline, param_dist, n_iter=20, random_state=0)
search.fit(train[features], train[label])
search.best_params_, f"{search.best_score_:.3f}"
```

Out[ ]:
```
({'logisticregression__tol': 0.01,
  'logisticregression__solver': 'lbfgs',
  'logisticregression__penalty': 'l2',
  'logisticregression__max_iter': 300,
  'logisticregression__C': 0.1,
  'columntransformer__kbinsdiscretizer__strategy': 'quantile',
  'columntransformer__kbinsdiscretizer__n_bins': 10,
  'columntransformer__kbinsdiscretizer__encode': 'onehot-dense'},
 '0.729')
```

Model Evaluation

สร้าง Baseline มาเปรียบเทียบกับโมเดลที่มีอยู่

In [ ]:
```python
# Dummy ยังไม่ได้ทำ feature engineer
from sklearn.dummy import DummyClassifier

# baseline = DummyClassifier(strategy="uniform", random_state=1234)
# baseline = DummyClassifier(strategy="stratified", random_state=1234)
baseline = DummyClassifier(strategy="most_frequent", random_state=1234)
# baseline = DummyClassifier(strategy="constant", constant=0, random_state=1234)

baseline.fit(train[features], train.Diabetes_binary)

baseline_train_score = baseline.score(train[features], train.Diabetes_binary)
baseline_validation_score = baseline.score(validation[features], validation.Diabetes_binary)

logistic_tuning_train_score = search.score(train[features], train[label])
logistic_tuning_validation_score = search.score(validation[features], validation[label])

print(f"Baseline Train Score: {baseline_train_score:.3f}\nBaseline Validation Score: {baseline_validation_score:.3f}")
print(f"Logistic Train Score: {logistic_tuning_train_score:.3f}\nLogistic Validation Score: {logistic_tuning_validation_score:.3f}")
```

```
Baseline Train Score: 0.500
Baseline Validation Score: 0.500
Logistic Train Score: 0.728
Logistic Validation Score: 0.730
```

Classification Report

In [ ]:
```python
from sklearn.metrics import classification_report

y_pred = search.best_estimator_.predict(validation[features])
print(classification_report(validation[label], y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.74      0.70      0.72      3159
         1.0       0.72      0.75      0.74      3159

    accuracy                           0.73      6318
   macro avg       0.73      0.73      0.73      6318
weighted avg       0.73      0.73      0.73      6318
```

Cross Validation

In [ ]:
```python
from sklearn.model_selection import cross_val_score

cross_val_logistic_pipeline = create_pipline_logistic()
cross_val_logistic_pipeline.set_params(**search.best_params_)

%time cv_scores = cross_val_score(cross_val_logistic_pipeline, train[features], train[label], cv = 5)
print(f"cross validation score: %.3f" % cv_scores.mean())
```

```
CPU times: total: 125 ms
Wall time: 258 ms
cross validation score: 0.729
```

Model Calibration

In [ ]:
```python
from sklearn.calibration import CalibratedClassifierCV

calibration_logistic_pipeline = create_pipline_logistic()
calibration_logistic_pipeline.set_params(**search.best_params_)

calibration_logistic_pipeline = CalibratedClassifierCV(estimator=calibration_logistic_pipeline, cv = 5)
%time calibration_logistic_pipeline.fit(train[features], train[label])

train_score = calibration_logistic_pipeline.score(train[features], train[label])
valid_score = calibration_logistic_pipeline.score(validation[features], validation[label])
test_score = calibration_logistic_pipeline.score(test[features], test[label])

print(f"Train Score: {train_score:.3f}")
print(f"Validation Score: {valid_score:.3f}")
print(f"Test Score: {test_score:.3f}")
```

```
CPU times: total: 109 ms
Wall time: 259 ms
Train Score: 0.730
Validation Score: 0.729
Test Score: 0.731
```

In [ ]:
```python
sample = test.sample(100)

cnt = 0

y_pred = calibration_logistic_pipeline.predict(sample[features])
sample_label = sample[label]
for i in range(len(sample)):
    if (y_pred[i] == sample_label.iloc[i]):
        cnt += 1

cnt
```

Out[ ]:
```
77
```

In [ ]:
```
%%writefile requirements.txt
pandas=2.2.2
scikit-learn=1.5.1
imbalanced-learn=0.12.3
gradio=4.42.0
```

```
Overwriting requirements.txt
```

In [ ]:
```python
test[features]
```

Out[ ]:

	GenHlth	HighBP	HighChol	BMI	Age	DiffWalk
10943	3.0	1.0	1.0	37.0	8.0	0.0
207394	3.0	1.0	0.0	26.0	8.0	0.0
67678	2.0	0.0	0.0	24.0	5.0	0.0
121461	4.0	1.0	1.0	28.0	13.0	1.0
77625	2.0	0.0	0.0	25.0	1.0	0.0
...	...	...	...	...	...	...
153876	3.0	0.0	0.0	24.0	13.0	0.0
71327	4.0	1.0	1.0	32.0	11.0	1.0
48017	2.0	0.0	1.0	24.0	8.0	0.0
207480	4.0	1.0	1.0	24.0	10.0	0.0
236197	4.0	1.0	1.0	28.0	7.0	0.0

14741 rows × 6 columns

Export "Artifacts"

In [ ]:
```python
import joblib
joblib.dump(calibration_logistic_pipeline, "pipeline.joblib")
```

Out[ ]:
```
['pipeline.joblib']
```

In [ ]:
```python
%%writefile app.py
import pandas as pd
import gradio as gr
import joblib
import numpy as np

features = ["GenHlth", "BMI", "HeartDiseaseorAttack", "DiffWalk", "HvyAlcoholConsump"]
pipeline = joblib.load("pipeline.joblib")
classes = ["จากการประเมินเบื้องต้นคุณไม่มีความเสี่ยงที่จะเป็นโรคเบาหวาน", "จากการประเมินเบื้องต้นคุณมีความเสี่ยงที่จะเป็นโรคเบาหวาน"]
def predict(genhlth, highbp, highchol, bmi, age, diffWalk):
  sample = dict()
  sample["GenHlth"] = genhlth
  sample["HighBP"] = highbp
  sample["HighChol"] = highchol
  sample["BMI"] = bmi
  sample["Age"] = age
  sample["DiffWalk"] = diffWalk

  sample = pd.DataFrame([sample])
  y_pred = pipeline.predict_proba(sample)[0]
  y_pred = dict(zip(classes, y_pred))
  return y_pred

with gr.Blocks() as demo:

  options_age = ["18-24", "25-29", "30-34", "35-39", "40-44", "45-49", "50-54", "55-59", "60-64", "65-69", "70-74", "75-79", ">80"]
  values_age = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13]
  options_bmi = ["13-23", "24-26", "27-29", "30-34", ">35"]
  values_bmi = [1, 2, 3 ,4, 5]
  options_yes_no = ["Yes", "No"]
  values_yes_no = [1, 0]

  genhlth = gr.Slider(value=1, label="General Health", info="Would you say that in general your health is: scale 1-5", minimum=1, maximum=5, step=1, interactive=True)
  with gr.Row():
    highbp = gr.Radio(choices=list(zip(options_yes_no, values_yes_no)), label="High Blood Pressure", info="Do you have High Blood Pressure?")
    highchol = gr.Radio(choices=list(zip(options_yes_no, values_yes_no)), label="High Cholesterol ", info="Do you have high cholesterol?")
  bmi = gr.Radio(choices=list(zip(options_bmi, values_bmi)), label="BMI", info="Body Mass Index")
  age = gr.Radio(choices=list(zip(options_age, values_age)), label="Age", info="Select your age")
  diffWalk = gr.Radio(choices=list(zip(options_yes_no, values_yes_no)), label="Difficult walk", info="Do you have serious difficulty walking or climbing stairs?")

  predict_btn = gr.Button("Predict", variant="primary")
  Diabetes_binary = gr.Label(label="Diabetes_binary")

  inputs = [genhlth, highbp, highchol, bmi, age, diffWalk]
  output = [Diabetes_binary]

  predict_btn.click(predict, inputs=inputs, outputs=output)
if __name__ == "__main__":
  demo.launch()
```

```
Overwriting app.py
```

In [ ]:
```python
%run app.py
```

```
Running on local URL:  http://127.0.0.1:7860

To create a public link, set `share=True` in `launch()`.
```

ERROR

The requested URL could not be retrieved

The following error was encountered while trying to retrieve the URL: http://127.0.0.1:7860/

Access Denied.

Access control configuration prevents your request from being allowed at this time. Please contact your service provider if you feel this is incorrect.

Your cache administrator is webmaster.

Generated Mon, 26 Aug 2024 10:52:00 GMT by htmlpdf-proxy-us (squid/4.10)

ลิงค์ไปยัง Hugging Face Space ของตัวเองที่ออกแบบให้เหมาะสมกับวัตถุประสงค์

Link to Hugging Face Space : https://huggingface.co/spaces/Chaiyawatt/diabetes-indicators

การมีส่วนร่วมของสมาชิกแต่ละคนในกลุ่ม (แต่ละคนทำอะไรบ้าง)

- นายรักษิต รุ่งรัตนไชย สร้างหน้าต่าง UI เพื่อให้ผู้ใช้ใช้งานโมเดลได้ และนำไป deploy ต่อไป
- นายบวรรัตน์ ตั้งนราธัชกิจ ทำกระบวนการต่างๆกับชุดข้อมูลเช่น data eng, training data, feature eng, model dev เป็นต้น เพื่อให้ได้โมเดลที่พร้อมทำงาน
- นายไชยวัตน์ หนูวัฒนา ทำการ deploy model ของเราขึ้นสู่ huggin face

การเปิดเผยการใช้เครื่องมือปัญญาประดิษฐ์ (ใช้อะไร ใช้เพื่ออะไร ใช้อย่างไร, prompt อย่างไร)

การทำงานในครั้งนี้มีการใช้ Chat GPT เพื่อค้นหาวิธีการใช้งาน components ของ Gradio, Prompt : ภายใน Gradio หากอยากให้ปุ่มกด radio คืนค่าเป็นค่าที่กำหนดทำอย่างไร