Enhancing AI Equity in Education: Post Implementation Development

Overview

In this notebook, we delve into crucial aspects of AI application in education, focusing on explainability, customer feedback and UX testing, and post-launch assessment. Our goal is to ensure that AI tools not only support educational objectives but also do so transparently and equitably, incorporating user feedback and continuously monitoring performance and bias.

Topics Covered

1. **AI Explainability, Literacy & Education**: Understanding the "why" behind AI predictions is crucial for trust and usability in educational settings. We'll explore tools and techniques for enhancing AI explainability.
2. **Customer Feedback and UX Testing**: Direct input from users is invaluable. We'll examine how to collect and use feedback to improve AI tools in education.
3. **Post-Launch Assessment**: After deployment, the work isn't done. We'll cover strategies for monitoring AI tools to detect any performance issues or bias, ensuring they remain effective and fair over time.

AI Explainability, Literacy & Education

AI literacy and end-user education promote the explainability of algorithms by increasing stakeholders' understanding of AI concepts and empowering end-users to engage with AI systems effectively.

Model Explainability

Model explainability is specific to the AI system and refers to the ability to provide transparent insights into its decision-making processes, enabling stakeholders, including end-users, to understand and interpret the rationale behind its outputs.

In [ ]:
```
!pip install shap
```

```
Collecting shap
  Downloading shap-0.44.1-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.many
linux_2_17_x86_64.manylinux2014_x86_64.whl (535 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 535.7/535.7 kB 6.7 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from
shap) (1.25.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from
shap) (1.11.4)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-package
s (from shap) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (fro
m shap) (1.5.3)
Requirement already satisfied: tqdm>=4.27.0 in /usr/local/lib/python3.10/dist-package
s (from shap) (4.66.2)
Requirement already satisfied: packaging>20.9 in /usr/local/lib/python3.10/dist-packa
ges (from shap) (23.2)
Collecting slicer==0.0.7 (from shap)
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied: numba in /usr/local/lib/python3.10/dist-packages (from
shap) (0.58.1)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages
(from shap) (2.2.1)
Requirement already satisfied: llvmlite<0.42,>=0.41.0dev0 in /usr/local/lib/python3.1
0/dist-packages (from numba->shap) (0.41.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/di
st-packages (from pandas->shap) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-package
s (from pandas->shap) (2023.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packag
es (from scikit-learn->shap) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist
-packages (from scikit-learn->shap) (3.3.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (f
rom python-dateutil>=2.8.1->pandas->shap) (1.16.0)
Installing collected packages: slicer, shap
Successfully installed shap-0.44.1 slicer-0.0.7
```

In [ ]:
```python
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.datasets import make_classification
import shap  # For explainability
import matplotlib.pyplot as plt
```

In [ ]:
```python
# Generate a synthetic dataset: features could represent student demographics, per
X, y = make_classification(n_samples=1000, n_features=20, n_informative=15, n_redundar

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=

print("Synthetic dataset generated and split into training and testing sets.")
```

```
Synthetic dataset generated and split into training and testing sets.
```

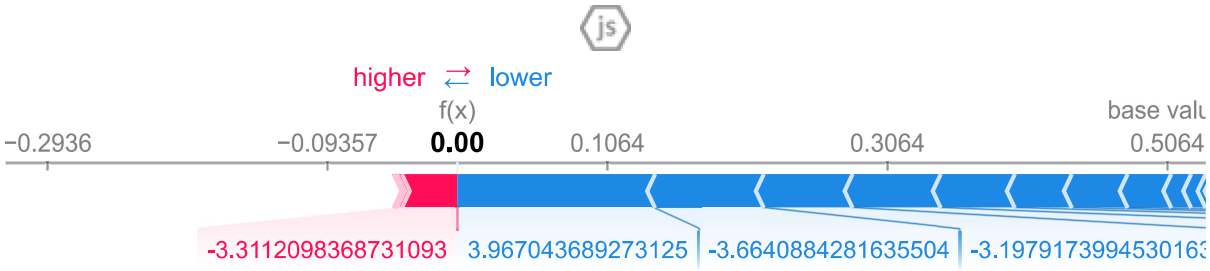
In [ ]:
```python
# Initialize and train a RandomForestClassifier
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)
```

```
# Predict on the test set
y_pred = model.predict(X_test)

# Basic evaluation
accuracy = accuracy_score(y_test, y_pred)
print(f"Model Accuracy: {accuracy:.2f}")
```

```
Model Accuracy: 0.98
```

In [ ]:
```
# Using SHAP for explainability
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_test)

# Plot the SHAP values for the first prediction
shap.initjs()
shap.force_plot(explainer.expected_value[1], shap_values[1][0,:], X_test[0,:])
```

Out[ ]:



Customer Feedback and UX Testing

Gathering and integrating customer feedback is essential for creating user-centered AI tools in education. Tools like surveys, interviews, and usability tests help understand user needs and preferences. Incorporating this feedback into the AI development process ensures the tool is effective and user-friendly.

Tools for Customer Feedback and UX Testing:

- **Surveys and Questionnaires**: Collect quantitative and qualitative data from users.
- **Interviews**: Deep dive into user experiences and expectations.
- **Usability Testing**: Observe real users interacting with the AI tool to identify usability issues.

In [ ]:
```
# Performance and Bias Detection
# Assuming y_test has equal representation of classes for simplicity

# Confusion matrix to detect bias in predictions
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", conf_matrix)

# Performance monitoring could be done through logging and regular evaluation against
```

```
Confusion Matrix:
 [[104   0]
 [  4  92]]
```

Post Launch Assessment Performance and Bias Detection

Ensuring the long-term success of an AI tool in education requires continuous monitoring for performance, fairness, and user satisfaction. Below are resources and tools that can help practitioners in these areas:

Regular Evaluation

- **Scikit-learn**: A foundational Python library that provides simple and efficient tools for data analysis and modeling, including performance metrics.
  - GitHub: https://github.com/scikit-learn/scikit-learn

Bias Detection

- **AI Fairness 360 (AIF360)**: An extensible open-source library to help detect, understand, and mitigate bias in machine learning models.
  - GitHub: https://github.com/Trusted-AI/AIF360
- **Fairlearn**: A toolkit that aims to empower data scientists and developers to assess and improve the fairness of their AI systems.
  - GitHub: https://github.com/fairlearn/fairlearn

User Feedback Loop

- **Prodigy**: An annotation tool for machine learning and natural language processing tasks. It's an efficient way to improve models based on user feedback, though it's not open source.
  - Website: https://prodi.gy/
- **Paperform**: A versatile form builder that can be used to collect user feedback, with easy integration into websites and apps.
  - Website: https://paperform.co/

While Prodigy and Paperform are not hosted on GitHub and might not be free, they represent practical tools for collecting and incorporating user feedback into the continuous improvement cycle of AI tools. For open-source alternatives, consider integrating GitHub issues or Google Forms as part of your feedback loop strategy.

In [1]:
```
import pandas as pd
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import Pipeline
import numpy as np
```

```
import pickle
from sklearn import preprocessing
import matplotlib.pyplot as plt
plt.rc("font", size=14)
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.utils import class_weight
from sklearn.metrics import balanced_accuracy_score
from sklearn.ensemble import IsolationForest
import seaborn as sns
sns.set(style="white")
sns.set(style="whitegrid", color_codes=True)
```

This dataset is an NHIS dataset showing the lifestyle factors and social determinants of health that impact the risk of pre-diabetes of ~27000 individuals.

In [3]:
```
df_NHIS_all_year = pd.read_csv('df_NHIS_data_filt2d_all_year_binned.csv')
df_NHIS_all_year = df_NHIS_all_year.drop(columns = 'Unnamed: 0')
df_NHIS_all_year.head()
```

Out[3]:

	SEX_A	AGEP_A	EDUC_A	RACEALLP_A	PCNTFAM_A	OVER65FLG_A	PHSTAT_A	HYPEV_A	CHLEV_A
0	1	85	5	1	2	1	2	2	
1	1	39	5	1	3	0	2	1	
2	1	42	4	2	1	0	3	2	
3	1	32	8	1	2	0	1	2	
4	2	85	4	1	1	1	3	1	

5 rows × 208 columns

In [4]:
```
var_list = ['AGEP_A','BIN_SEX_A',
 'BIN_HYPEV_A',
 'BIN_CHLEV_A',
 'BIN_CHDEV_A',
 'BIN_STREV_A',
 'BIN_ASEV_A',
 'BIN_CANEV_A',
 'BIN_GESDIB_A',
 'BIN_DIBEV_A',
 'BIN_DIBREL_A',
 'BIN_COPDEV_A',
 'BIN_ARTHEV_A',
 'BIN_DEMENEV_A',
 'BIN_ANXEV_A',
 'BIN_DEPEV_A',
 'BIN_PAYBLL12M_A',
 'BIN_DENNG12M_A',
 'BIN_HOSPONGT_A',
 'BIN_MEDNG12M_A',
 'BIN_RXDL12M_A',
 'BIN_SHTFLU12M_A',
 'BIN_EYEEX12M_A',
 'BIN_SMKEV_A',
```

```
 'BIN_FSNAP12M_A',
 'BIN_FWIC12M_A',
 'BIN8_EDUC_A',
 'OVER65FLG_A',
 'BIN3_PHSTAT_A',
 'BIN3_VISIONDF_A',
 'BIN3_HEARINGDF_A',
 'BIN3_DIFF_A',
 'BIN3_COMDIFF_A',
 'BIN3_COGMEMDFF_A',
 'BIN3_UPPSLFCR_A',
 'BIN3_SOCSCLPAR_A',
 'BIN1_USPLKIND_A',
 'BIN4_EMERG12MTC_A',
 'BIN2_PAIFRQ3M_A',
 'BIN1_DRK12MWK_A',
 'BIN2_MODFREQW_A',
 'BIN6_SLPHOURS_A',
 'BIN1_SLPREST_A',
 'BIN_MARSTAT_A',
 'BIN_EMPRSNOWK_A',
 'BIN_NPOV200',
 'BIN2_FDSBALANCE_A',
 'BIN_HOUTENURE_A',
'BIN_ANGEV_A',
'BIN_MIEV_A',
'BIN_PREDIB_A',
'BIN_DENDL12M_A',
'BIN_MEDDL12M_A',
'BIN_RXDG12M_A',
'BIN1_FDSRUNOUT_A',
'BIN2_FDSLAST_A',
'BIN2_PAYWORRY_A',
'BIN_FDSSKIP_A',
'BIN_RACEALLP_A',
'ADA_FULL_SCORE1']
```

In [5]:
```
df_nhis_filt = df_NHIS_all_year[var_list]
print(df_nhis_filt.shape)
df_nhis_filt.head()
```

```
(26836, 60)
```

Out[5]:

	AGEP_A	BIN_SEX_A	BIN_HYPEV_A	BIN_CHLEV_A	BIN_CHDEV_A	BIN_STREV_A	BIN_ASEV_A	BIN_
0	85	1	0	0	0	0	0	
1	39	1	1	0	0	0	0	
2	42	1	0	1	0	0	0	
3	32	1	0	0	0	0	0	
4	85	0	1	0	0	0	0	

5 rows × 60 columns

In [6]:
```
X = df_nhis_filt.values
```

In [7]:
```
#Implement Isolation Forest on Data to detect anomalies
clf = IsolationForest(random_state=0).fit(X)
```

In [8]:
```
isolation_scores = clf.predict(X)
```

In [9]:
```
isolation_scores
```

Out[9]:
```
array([-1,  1,  1, ...,  1,  1,  1])
```

In [10]:
```
df_nhis_filt['Isolation_Score']= isolation_scores
```

```
<ipython-input-10-0a20bef04240>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_nhis_filt['Isolation_Score']= isolation_scores
```

In [11]:
```
df_nhis_filt.head()
```

Out[11]:

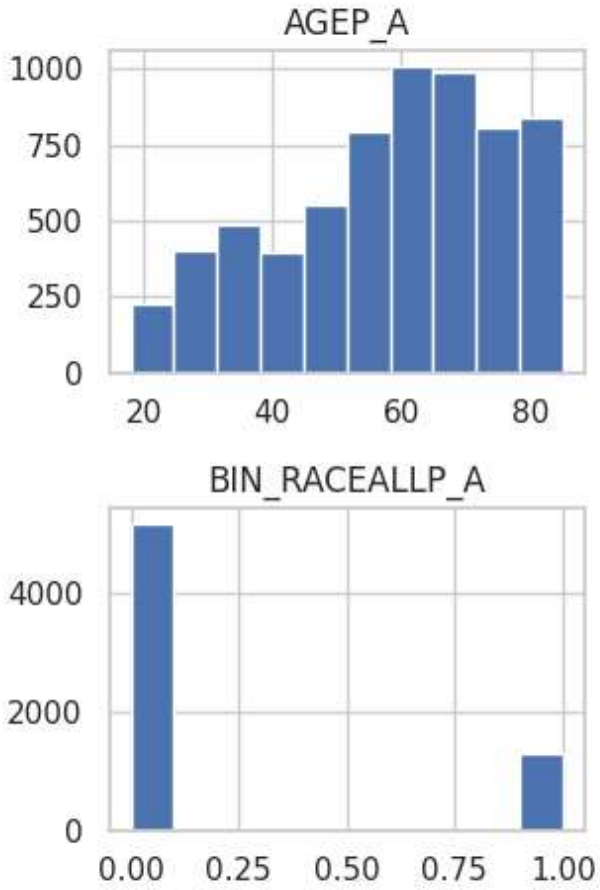
	AGEP_A	BIN_SEX_A	BIN_HYPEV_A	BIN_CHLEV_A	BIN_CHDEV_A	BIN_STREV_A	BIN_ASEV_A	BIN_
0	85	1	0	0	0	0	0	
1	39	1	1	0	0	0	0	
2	42	1	0	1	0	0	0	
3	32	1	0	0	0	0	0	
4	85	0	1	0	0	0	0	

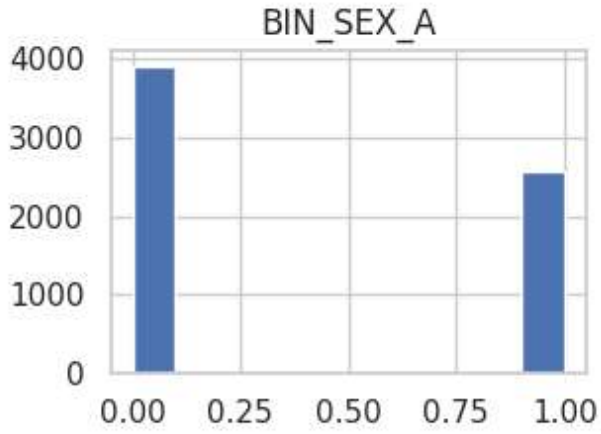
5 rows × 61 columns

In [12]:
```
df_anomaly = df_nhis_filt[df_nhis_filt['Isolation_Score']==-1]
```

The code piece below shows the data points that are identified as anomalies using the Isolation Forest code. BIN_SEX_A = 1 indicates male and 0 female. BIN_RACEALLP_A = 1 indicates Black, Latino, Pacific Islander and 0 represents white or caucasian. The anomaly detection algorithm shows that older (>60) black and white individuals are outliers. Taking this data out of the training set used to train an ML model to predict the risk of pre-diabetes will disadvantage this demographic group. Therefore, to ensure equitable outcomes, a separate ML model can be trained for this demographic.

In [13]:
```
df_anomaly[['AGEP_A','BIN_SEX_A','BIN_RACEALLP_A']].hist()
plt.tight_layout()
```


AGEP_A
1000
750
500
250
0
20
40
60
80


BIN_SEX_A
4000
3000
2000
1000
0
0.00
0.25
0.50
0.75
1.00

BIN_RACEALLP_A
4000
2000
0
0.00
0.25
0.50
0.75
1.00