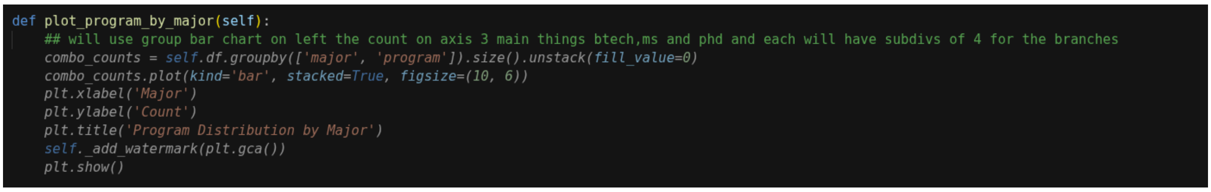
SMAI Assignment 1 prompts

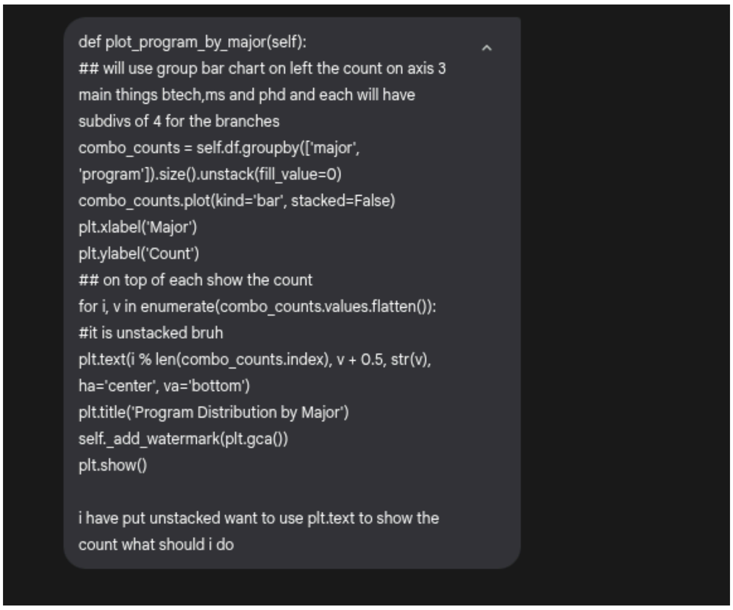
Q1

1. Learning and discussing abt plt - https://chatgpt.com/share/68a95c48-29f8-800f-b031-4ed1ee263950
2. Copilot for grouped bar charts



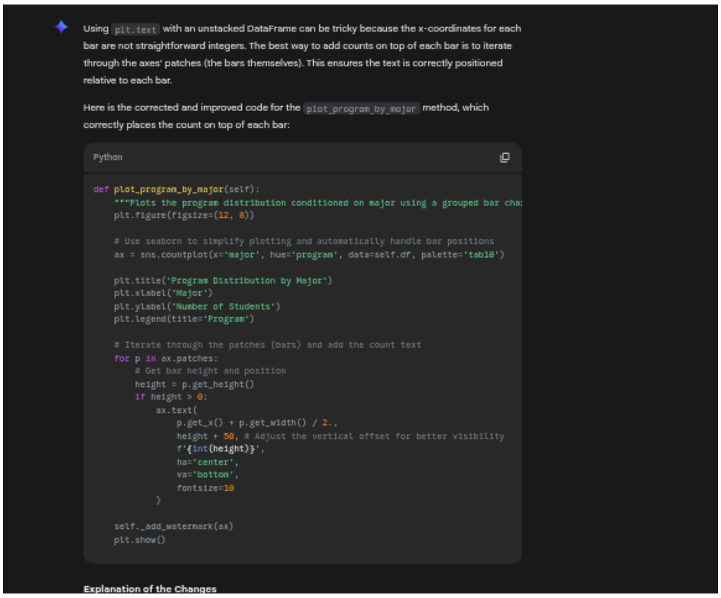
```python
def plot_program_by_major(self):
    ## will use group bar chart on left the count on axis 3 main things btech,ms and phd and each will have subdivs of 4 for the branches
    combo_counts = self.df.groupby(['major', 'program']).size().unstack(fill_value=0)
    combo_counts.plot(kind='bar', stacked=True, figsize=(10, 6))
    plt.xlabel('Major')
    plt.ylabel('Count')
    plt.title('Program Distribution by Major')
    self._add_watermark(plt.gca())
    plt.show()
```

Then gave this to gemini for improving



```
def plot_program_by_major(self):
## will use group bar chart on left the count on axis 3 main things btech,ms and phd and each will have subdivs of 4 for the branches
combo_counts = self.df.groupby(['major', 'program']).size().unstack(fill_value=0)
combo_counts.plot(kind='bar', stacked=False)
plt.xlabel('Major')
plt.ylabel('Count')
## on top of each show the count
for i, v in enumerate(combo_counts.values.flatten()):
#it is unstacked bruh
plt.text(i % len(combo_counts.index), v + 0.5, str(v), ha='center', va='bottom')
plt.title('Program Distribution by Major')
self._add_watermark(plt.gca())
plt.show()
```

i have put unstacked want to use plt.text to show the count what should i do



Using `plt.text` with an unstacked DataFrame can be tricky because the x-coordinates for each bar are not straightforward integers. The best way to add counts on top of each bar is to iterate through the axes' patches (the bars themselves). This ensures the text is correctly positioned relative to each bar.

Here is the corrected and improved code for the `plot_program_by_major` method, which correctly places the count on top of each bar:

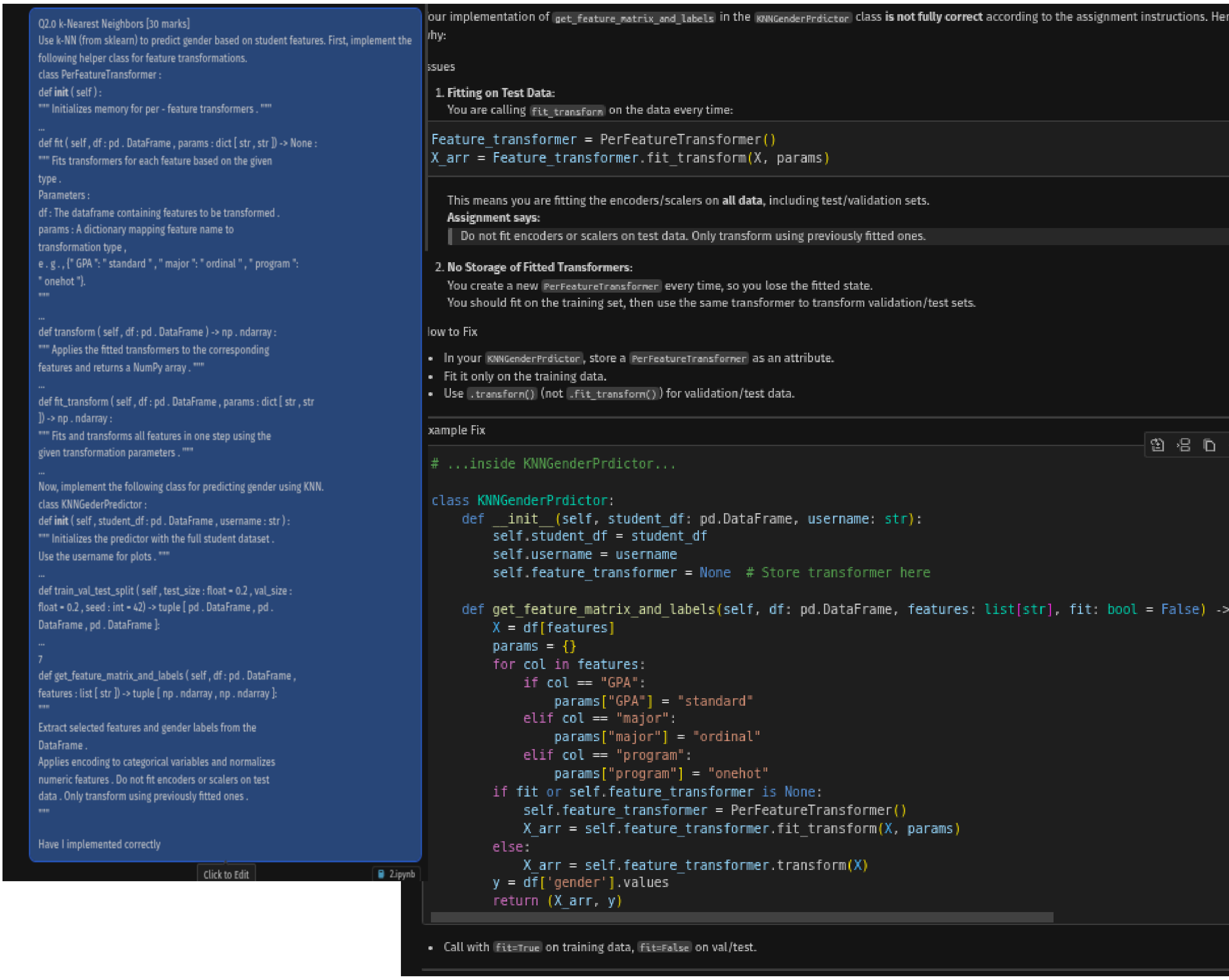
```python
def plot_program_by_major(self):
    """Plots the program distribution conditioned on major using a grouped bar cha
    plt.figure(figsize=(12, 8))

    # Use seaborn to simplify plotting and automatically handle bar positions
    ax = sns.countplot(x='major', hue='program', data=self.df, palette='tab10')

    plt.title('Program Distribution by Major')
    plt.xlabel('Major')
    plt.ylabel('Number of Students')
    plt.legend(title='Program')

    # Iterate through the patches (bars) and add the count text
    for p in ax.patches:
        # Get bar height and position
        height = p.get_height()
        if height > 0:
            ax.text(
                p.get_x() + p.get_width() / 2.,
                height + 50, # Adjust the vertical offset for better visibility
                f'{int(height)}',
                ha='center',
                va='bottom',
                fontsize=10
            )

    self._add_watermark(ax)
    plt.show()
```

Explanation of the Changes

3. Heatmap - https://chatgpt.com/share/68aa0039-ff54-800f-8e00-8e64196efadd
4. Stratified sampling doubts - https://chatgpt.com/share/68aa1a58-f770-800f-aa99-fbd3da28ec69
5. Understanding cohort and stuff - https://chatgpt.com/share/68aa9a50-4e10-800f-b518-983aec294dbb
6. Pairplots - https://claude.ai/share/cfb23927-5a61-4a25-b9d3-440788510134

Q2

1. Some stuff I did wrong as I did not understood so copilot



```
Q2.0 k-Nearest Neighbors [30 marks]
Use k-NN (from sklearn) to predict gender based on student features. First, implement the
following helper class for feature transformations.
class PerFeatureTransformer :
def init ( self ) :
""" Initializes memory for per - feature transformers . """
...
def fit ( self , df : pd . DataFrame , params : dict [ str , str ]) -> None :
""" Fits transformers for each feature based on the given
type .
Parameters :
df : The dataframe containing features to be transformed .
params : A dictionary mapping feature name to
transformation type ,
e . g . ,{" GPA ": " standard " , " major ": " ordinal " , " program ":
" onehot "}.
"""

...
def transform ( self , df : pd . DataFrame ) -> np . ndarray :
""" Applies the fitted transformers to the corresponding
features and returns a NumPy array . """
...
def fit_transform ( self , df : pd . DataFrame , params : dict [ str , str
]) -> np . ndarray :
""" Fits and transforms all features in one step using the
given transformation parameters . """
...
Now, implement the following class for predicting gender using KNN.
class KNNGederPredictor :
def init ( self , student_df : pd . DataFrame , username : str ) :
""" Initializes the predictor with the full student dataset .
Use the username for plots . """
...
def train_val_test_split ( self , test_size : float = 0.2 , val_size :
float = 0.2 , seed : int = 42) -> tuple [ pd . DataFrame , pd .
DataFrame , pd . DataFrame ]:
...
7
def get_feature_matrix_and_labels ( self , df : pd . DataFrame ,
features : list [ str ]) -> tuple [ np . ndarray , np . ndarray ]:
"""
Extract selected features and gender labels from the
DataFrame .
Applies encoding to categorical variables and normalizes
numeric features . Do not fit encoders or scalers on test
data . Only transform using previously fitted ones .
"""

Have I implemented correctly
```

our implementation of `get_feature_matrix_and_labels` in the `KNNGenderPrdictor` class **is not fully correct** according to the assignment instructions. Her
hy:

ssues

1. **Fitting on Test Data:**
   You are calling `fit_transform` on the data every time:

```
Feature_transformer = PerFeatureTransformer()
X_arr = Feature_transformer.fit_transform(X, params)
```

   This means you are fitting the encoders/scalers on **all data**, including test/validation sets.
   Assignment says:
   > Do not fit encoders or scalers on test data. Only transform using previously fitted ones.

2. **No Storage of Fitted Transformers:**
   You create a new `PerFeatureTransformer` every time, so you lose the fitted state.
   You should fit on the training set, then use the same transformer to transform validation/test sets.

low to Fix

- In your `KNNGenderPrdictor`, store a `PerFeatureTransformer` as an attribute.
- Fit it only on the training data.
- Use `.transform()` (not `.fit_transform()`) for validation/test data.

xample Fix

```
# ...inside KNNGenderPrdictor...

class KNNGenderPrdictor:
    def __init__(self, student_df: pd.DataFrame, username: str):
        self.student_df = student_df
        self.username = username
        self.feature_transformer = None  # Store transformer here

    def get_feature_matrix_and_labels(self, df: pd.DataFrame, features: list[str], fit: bool = False) ->
        X = df[features]
        params = {}
        for col in features:
            if col == "GPA":
                params["GPA"] = "standard"
            elif col == "major":
                params["major"] = "ordinal"
            elif col == "program":
                params["program"] = "onehot"
        if fit or self.feature_transformer is None:
            self.feature_transformer = PerFeatureTransformer()
            X_arr = self.feature_transformer.fit_transform(X, params)
        else:
            X_arr = self.feature_transformer.transform(X)
        y = df['gender'].values
        return (X_arr, y)
```

- Call with `fit=True` on training data, `fit=False` on val/test.

2.

Q3

1. Understanding if we need stdscaling for one hot encoding - https://chatgpt.com/share/68ad6ed5-ba74-800f-9d53-430eeb8792d5