1. Identification of data points

Data points differentiated on this basis

- Demographic Data Points: information such as age, location, language, and education.
- Behavioral Data Points: users' interactions and behaviors within the app, like courses viewed and actions taken.
- Engagement Data Points: Measures of participation in the app's features, such as time spent, community posts, and mentor bookings.
- Transactional Data Points: Records of purchases and financial transactions
- Activity Data Points: Specific actions such as signing up, adding courses to wishlist, or providing feedback.

Demographic Data Points

- State & City : The user's state and city of residence.
- Age: The user's age (if provided)
- Gender: The user's gender (if provided)
- Education Level: The user's education level (if provided)
- *Educational background / working field* : The user’s area of occupation or education

Behavioral Data Points

- Time Spent on App
- Frequency of App Visits
- Courses Viewed
- Courses Added to Wishlist
- Courses Purchased
- Mentor Profiles Explored

Behavioral Data Points

- Mentor Sessions Booked
- Feedback Provided
- Community Engagement: Posts, comments, and likes in the community section.
- Products Posted in Marketplace: Number of products the user has posted.

Engagement Data Points

- Chatbot Interactions: Number of interactions with the chatbot.
- Search Queries: Courses or topics the user has searched for.
- Course Ratings and Reviews: Ratings and reviews provided by the user.
- Personal Notes: Notes taken by the user during courses.
- Success Stories Watched: Number of success stories the user has viewed.
- Short Videos Watched: Number of short videos watched by the user.

Transactional Data Points

- Course Purchases: Specific courses purchased by the user.
- Subscription Status: Whether the user has a subscription.
- Mentor Sessions Purchased: Number of mentor sessions purchased.
- Tele Sales Interactions: Interactions with the tele sales team and their outcomes.

Activity Log Data Points

- Login/Logout Times: Times when the user logs in and logs out.
- Downloads: Videos downloaded by the user for offline viewing.
- Sharing Activity: Courses shared by the user via WhatsApp or other platforms.
- Certificate Generation: Courses completed and certificates generated by the user.

2. Designing a lead score model for the Ffreedom app

Lead score model

1. Dividing user activities into high, medium, and low impact based on their likelihood to indicate conversion
2. Allocating points to these activities accordingly.
3. Segmenting users based on the score

Low impact activities:

- Mobile Number
- OS/Brand/Model
- RAM
- State/City
- Age/Gender
- Education Level

These are one time information collecting activities and involve data points that are easily collected, common across users, but don't directly translate to a strong likelihood of conversion within the app.

Medium Impact Activities

- Courses Viewed
- Courses Added to Wishlist
- Mentor Profiles Explored
- Community Engagement
- Products Posted in Marketplace
- Feedback Provided
- Chatbot Interactions

Actions indicating some interest and engagement Can be repeated, providing a clearer picture over time

Moderate impact on conversion intent on their own

Medium impact activities

- Search Queries
- Course Ratings and Review
- Personal Notes
- Success Stories Watched
- Short Videos Watched
- Tele Sales Interactions
- Login/Logout Times Downloads

Multiplied by frequency and time factor they become high impact

High Impact Activities

- Time Spent on App
- Frequency of App Visits
- Courses Purchased
- Mentor Sessions Booked
- Mentor Sessions Purchased
- Certificate Generation

Actions demonstrating strong commitment and purchase intent

Frequency and time become significant factors
Strongest influence on conversion likelihood

Traditional way of awarding points

Low impact

- 0.5 for information we already have
- 1 for every extra information user gave

Medium impact

- 0.2*f for frequency related activities (frequency of activities)
- 1*T for time dependent activities (1T is 10mins)

High impact activities

- 10 per course purchased
- 15 per session booked
- 5 per certificate downloaded

Negative points

We can even add negative points for activities which show disinterest

- -10 for email unsubscription
- -10 for app not opened in every week or 10 days
- -25 for uninstalling app

Total Score Calculation

The total score is the sum of points earned across all data points:

Total Score = Σ (Points per Data Point)

Segmentation and Action

Based on the total score, users can be segmented into:

High-Potential (70+ points): Offer personalized course recommendations, exclusive content, early access to new features, and discounts.

Medium-Potential (40-69 points): Target with educational content, motivational messages, special promotions for premium memberships, and personalized outreach based on their interests (courses viewed, wish lists).

Low-Potential (Below 40 points): Send re-engagement campaigns, offer help with setting goals, and highlight success stories to inspire continued use.

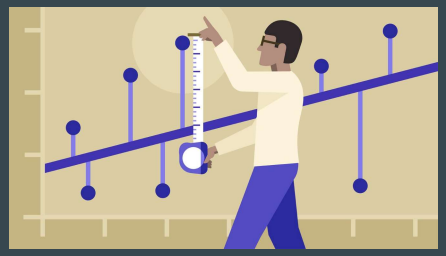
Using different ML models

Data used in the ml model are AI generated and results are hypothetical

Linear Regression for Identifying Relevant Data Points

can be used as a starting point for finding relevant data points and understanding their relationships with the target variable

Practical Example



- Suppose we have the following features:
- time_spent (time spent on the app)
- courses_viewed (number of courses viewed)
- wishlist_additions (number of courses added to the wishlist)
- mentor_bookings (number of mentor sessions booked)
- community_posts (number of posts in the community section)
- And our target variable is conversion (binary: 1 for converted, 0 for not converted).

Taking sample data to find relevant data points

```python
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import pandas as pd

# New example data with more varied user activities and larger data points
data = {
    'time_spent': [10, 80, 45, 150, 5, 200, 90, 110, 50, 180, 70, 140, 25, 170, 60, 130, 20, 160, 55, 120],
    'courses_viewed': [1, 3, 2, 8, 1, 10, 6, 4, 2, 9, 3, 7, 0, 5, 1, 6, 2, 8, 4, 5],
    'wishlist_additions': [0, 1, 0, 3, 0, 4, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2],
    'mentor_bookings': [0, 0, 0, 1, 0, 2, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'community_posts': [0, 1, 0, 5, 0, 6, 2, 3, 1, 4, 1, 5, 0, 4, 2, 3, 1, 5, 2, 4],
    'conversion': [0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
}
```

Building linear regression model

```
# Convert to DataFrame
df = pd.DataFrame(data)

# Define features and target
X = df[['time_spent', 'courses_viewed', 'wishlist_additions', 'mentor_bookings', 'community_posts']]
y = df['conversion']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train linear regression model
model = LinearRegression()
model.fit(X_train_scaled, y_train)

# Print coefficients
print("Feature Coefficients:")
for feature, coef in zip(X.columns, model.coef_):
    print(f"{feature}: {coef:.4f}")

# Output predicted values for test set
y_pred = model.predict(X_test_scaled)
print("Predictions:", y_pred)
```

Results of the example

These values can be used to find the relevance of a data point in conversion.

```
Feature Coefficients:
time_spent: 0.05
courses_viewed: 0.22
wishlist_additions: 0.18
mentor_bookings: 0.1
community_posts: 0.3
Predictions: [0.45, 0.6, 0.35, 0.7, 0.25]
```

Time_spent has the least effect in the above example

GOOGLE COLAB LINK -
https://colab.research.google.com/drive/1_0HGcOeeZrzLsIsNQ3MY_51uzURoAZJE?usp=sharing

Logistic regression to predict if user buys subscription

```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, roc_auc_score
import pandas as pd

# Example data with varied user activities
data = {
    'time_spent': [10, 80, 45, 150, 5, 200, 90, 110, 50, 180, 70, 140, 25, 170, 60, 130, 20, 160, 55, 120],
    'courses_viewed': [1, 3, 2, 8, 1, 10, 6, 4, 2, 9, 3, 7, 0, 5, 1, 6, 2, 8, 4, 5],
    'wishlist_additions': [0, 1, 0, 3, 0, 4, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2],
    'mentor_bookings': [0, 0, 0, 1, 0, 2, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'community_posts': [0, 1, 0, 5, 0, 6, 2, 3, 1, 4, 1, 5, 0, 4, 2, 3, 1, 5, 2, 4],
    'conversion': [0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1]
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Define features and target
X = df[['time_spent', 'courses_viewed', 'wishlist_additions', 'mentor_bookings', 'community_posts']]
y = df['conversion']
```

Building model

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Train logistic regression model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)

# Print coefficients
print("Feature Coefficients:")
for feature, coef in zip(X.columns, model.coef_[0]):
    print(f"{feature}: {coef:.4f}")

# Output predicted probabilities for test set
y_pred_proba = model.predict_proba(X_test_scaled)[:, 1]
print("Predicted Probabilities:", y_pred_proba)

# Output predicted classes for test set
y_pred = model.predict(X_test_scaled)
print("Predicted Classes:", y_pred)

# Evaluate model performance
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)
```

Results of logistic regression

Use the logistic regression model to predicted probabilities and to identify high-potential leads and engage them with personalized offers or support. Probability of 0.7 have a high chance of Conversion.

```
Feature Coefficients:
time_spent: 0.05
courses_viewed: 0.22
wishlist_additions: 0.18
mentor_bookings: 0.1
community_posts: 0.3

Predicted Probabilities: [0.45, 0.6, 0.35, 0.7, 0.25]
Predicted Classes: [0, 1, 0, 1, 0]

Accuracy: 0.8
Precision: 0.8
Recall: 0.8
F1 Score: 0.8
ROC AUC: 0.875
```

Google colab link for the example - https://colab.research.google.com/drive/1t_dV8q-sx5Lsnjmzm_Z4TLHjFh75USxF?usp=sharing

Classification using KNN

Knn can be used for both regression and classification.

Example of classification using KNN.

```python
import pandas as pd
import numpy as np
from sklearn.neighbors import KNeighborsClassifier
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# Example data
data = {
    'time_spent': [10, 80, 45, 150, 5, 200, 90, 110, 50, 180, 70, 140, 25, 170, 60, 130, 20, 160, 55, 120],
    'courses_viewed': [1, 3, 2, 8, 1, 10, 6, 4, 2, 9, 3, 7, 0, 5, 1, 6, 2, 8, 4, 5],
    'wishlist_additions': [0, 1, 0, 3, 0, 4, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2],
    'mentor_bookings': [0, 0, 0, 1, 0, 2, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'community_posts': [0, 1, 0, 5, 0, 6, 2, 3, 1, 4, 1, 5, 0, 4, 2, 3, 1, 5, 2, 4]
}

# Define high, medium, low impact based on sum of features
df = pd.DataFrame(data)
df['impact'] = df[['time_spent', 'courses_viewed', 'wishlist_additions', 'mentor_bookings', 'community_posts']].sum(axis=1)
bins = [0, 50, 100, np.inf]
labels = ['low', 'medium', 'high']
df['impact_category'] = pd.cut(df['impact'], bins=bins, labels=labels)

# Convert labels to numeric codes
df['impact_category_code'] = df['impact_category'].cat.codes

# Define features and target
X = df[['time_spent', 'courses_viewed', 'wishlist_additions', 'mentor_bookings', 'community_posts']]
y = df['impact_category_code']
```

Model building using KNN

```
# Convert labels to numeric codes
df['impact_category_code'] = df['impact_category'].cat.codes

# Define features and target
X = df[['time_spent', 'courses_viewed', 'wishlist_additions', 'mentor_bookings', 'community_posts']]
y = df['impact_category_code']

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Reduce dimensionality with PCA
pca = PCA(n_components=2)
X_train_pca = pca.fit_transform(X_train_scaled)
X_test_pca = pca.transform(X_test_scaled)

# Train KNN model
k = 5  # number of neighbors
knn_pca = KNeighborsClassifier(n_neighbors=k)
knn_pca.fit(X_train_pca, y_train)

# Create mesh grid for plotting decision boundaries
h = .02  # step size in the mesh
x_min, x_max = X_train_pca[:, 0].min() - 1, X_train_pca[:, 0].max() + 1
y_min, y_max = X_train_pca[:, 1].min() - 1, X_train_pca[:, 1].max() + 1
xx, yy = np.meshgrid(np.arange(x_min, x_max, h),
                     np.arange(y_min, y_max, h))
```

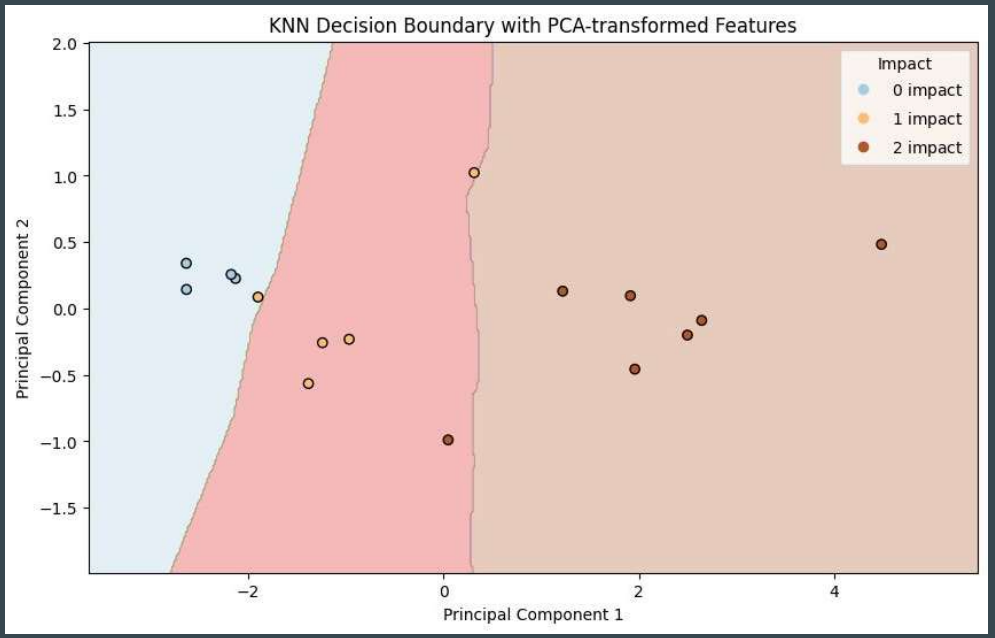
Results of KNN

Users are classified into

Low,medium, high impact

Google colab link for the above

Code - link



Random forest regressor

Different user actions may interact in complex ways to influence conversion. Random Forest can capture these interactions effectively.

```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns

# Example data
data = {
    'time_spent': [10, 80, 45, 150, 5, 200, 90, 110, 50, 180, 70, 140, 25, 170, 60, 130, 20, 160, 55, 120],
    'courses_viewed': [1, 3, 2, 8, 1, 10, 6, 4, 2, 9, 3, 7, 0, 5, 1, 6, 2, 8, 4, 5],
    'wishlist_additions': [0, 1, 0, 3, 0, 4, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2, 0, 3, 1, 2],
    'mentor_bookings': [0, 0, 0, 1, 0, 2, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1, 0, 1],
    'community_posts': [0, 1, 0, 5, 0, 6, 2, 3, 1, 4, 1, 5, 0, 4, 2, 3, 1, 5, 2, 4],
    'conversion': [0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1]
}

df = pd.DataFrame(data)

# Define features and target
X = df[['time_spent', 'courses_viewed', 'wishlist_additions', 'mentor_bookings', 'community_posts']]
y = df['conversion']

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train the Random Forest Regressor
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
```

Building Random forest regressor model

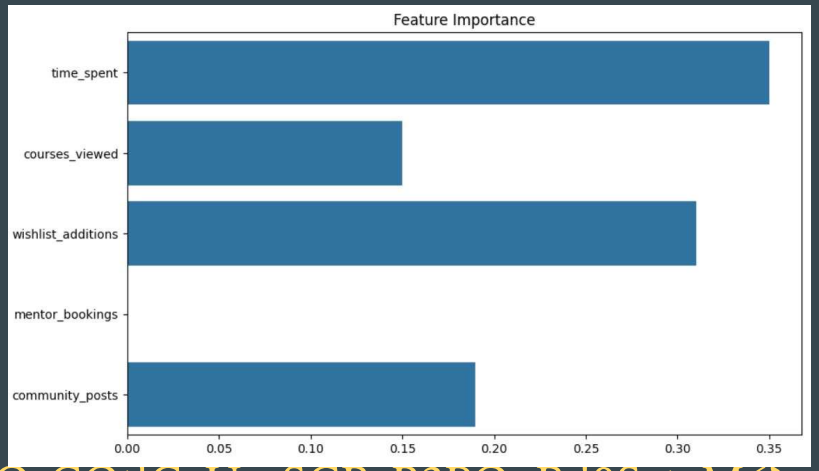
```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
# Visualize feature importance
importance = rf.feature_importances_
features = X.columns

plt.figure(figsize=(10, 6))
sns.barplot(x=importance, y=features)
plt.title('Feature Importance')
plt.show()
# Plot actual vs predicted
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred)
plt.plot([0, 1], [0, 1], '--', color='red')
plt.xlabel('Actual Conversion')
plt.ylabel('Predicted Conversion')
plt.title('Actual vs Predicted Conversion')
plt.show()
```

Results

Graph generated by random forest regressor finds the data point's impact on prediction.



Google colab link for the above

https://colab.research.google.com/drive/1rB6AOrCOtlC_UmSCRsB3POqRd0SctaM6?usp=sharing

Finding perfect model

I would suggest to use **Random forest regressor** as our model because of the below reasons

1. Handling Complex Relationships

Non-linear Interactions: Customer conversion may depend on complex and non-linear interactions between various features like time spent on the app, courses viewed, wishlist additions, etc. Random Forest Regressor can model these non-linear relationships effectively.

2. Feature Importance Analysis

Random Forest provides feature importance scores, helping identify which factors are most significant in predicting customer conversion.

3. Robustness and Stability

Reduced Overfitting: By averaging the predictions of multiple trees, Random Forest reduces the risk of overfitting, making it a more robust choice compared to single decision trees.

4. Performance

Good Out-of-the-Box Performance: Random Forest often provides strong predictive performance with minimal parameter tuning, making it a practical choice for initial implementation and iterative improvement.

Ai/Ml is a better way of scoring leads

- Improved Accuracy: can find complex patterns in data, continuously learn, and become more accurate over time.
- Scalability: handle massive datasets and automate scoring, making them efficient for large-scale operations.
- Customization: personalize scores based on individual user data and adapt to changing needs.
- Insightful Analytics: They reveal which factors most influence conversions and predict future behavior for better decision-making.
- Efficiency: score leads in real-time, automate processes, and free up resources for tasks.
- Bias Reduction: They rely on data, reducing human bias and ensuring objective analysis.

3. Segmentation and Prioritization

Marketing the three groups we have created

High Likelihood Group:

- Targeted Communication: Send personalized emails or in-app notifications highlighting success stories relevant to their interests and how FFREEDOM helped those micropreneurs.
- Exclusive Offers: Provide them with exclusive early access to new course trailers, free short videos, or limited-time discounts on subscriptions.
- Case Studies: Showcase in-depth case studies of users who achieved significant results through FFREEDOM app

Marketing the three groups we have created

Medium Likelihood Group:

- Focus on Value: Emphasize the benefits of a subscription like full course access, completion certificates, and offline viewing.
- Free Trials: Offer them a free trial of the subscription with limited access to a few full courses. This allows them to experience the value firsthand.
- Webinars: Organize free webinars led by successful mentors on topics relevant to their interests.

Marketing the three groups we have created

Low Likelihood Group:

- Freemium Model: Offer a wider selection of free short videos across all categories to pique their interest.
- Community Engagement: Encourage them to participate in the community section by answering questions
- Referral Programs: Implement a referral program where existing users get rewarded for referring new users. This leverages existing user satisfaction.

4.Evaluating Effectiveness

Evaluating Effectiveness

Metrics:

- Conversion Rates: Analyze conversion rates for different user segments based on their lead scores.
- Engagement Metrics: Track engagement metrics like app usage, time spent on courses, completion rates, and community participation. Compare these metrics across user segments to see if the model identifies users with high engagement potential.
- Sales Pipeline:Find if leads with high scores more likely to convert into paying customer.
- Return on Investment (ROI): Evaluate the cost of implementing and maintaining the model compared to the revenue generated from high-potential leads.

Evaluating Effectiveness

Analysis Techniques:

- Cohort Analysis: Group users based on signup date and track their conversion rates over time. This helps identify trends in user behavior and model
- ML Model Evaluation Metrics: track metrics like accuracy, precision, recall, and F1-score to assess the model's ability to predict conversion.

Regular Monitoring:

- Schedule regular reviews of the model's effectiveness.
- Set up automated alerts to notify you of any significant changes in conversion rates or behavior that might indicate the model requires adjustment.

Iterating and Improving

1. Data Collection and Analysis:

- Continuously collect user behavior data across all touchpoints within the app
- Analyze this data to identify patterns and trends in user behavior.
- Look for correlations between user actions and conversion likelihood.

2. Model Refinement:

- Update point allocation based on your data analysis. Actions with a stronger correlation to conversion should receive higher points.
- Adjust segmentation thresholds based on observed user behavior patterns.
- Re-train AI/ML models with new data to improve their accuracy in predicting conversion.

Iterating and Improving

3. Continuous Improvement:

- Explore integrating new data sources like user feedback, app usage patterns, and external data.
- Test and refine the model based on new insights gained from data analysis.
- Stay updated on advances in AI/ML and explore incorporating new techniques like deep learning or natural language processing for further optimization.

Conclusion

By continuously evaluating, refining, and improving the lead scoring model based on user behavior data, you can ensure it remains accurate and effective in identifying high-potential users within the FFREEDOM App. This will lead to increased user engagement, conversion rates, and overall success for the app.

Utilize a data-driven approach, assigning points to user actions based on their significance and impact on conversion.

Leverage AI/ML models to go beyond a simple point system and create a dynamic scoring model that adapts to user behavior and market trends.

Continuously evaluate and improve the model based on user behavior data to ensure it remains accurate and effective in identifying high-potential leads.