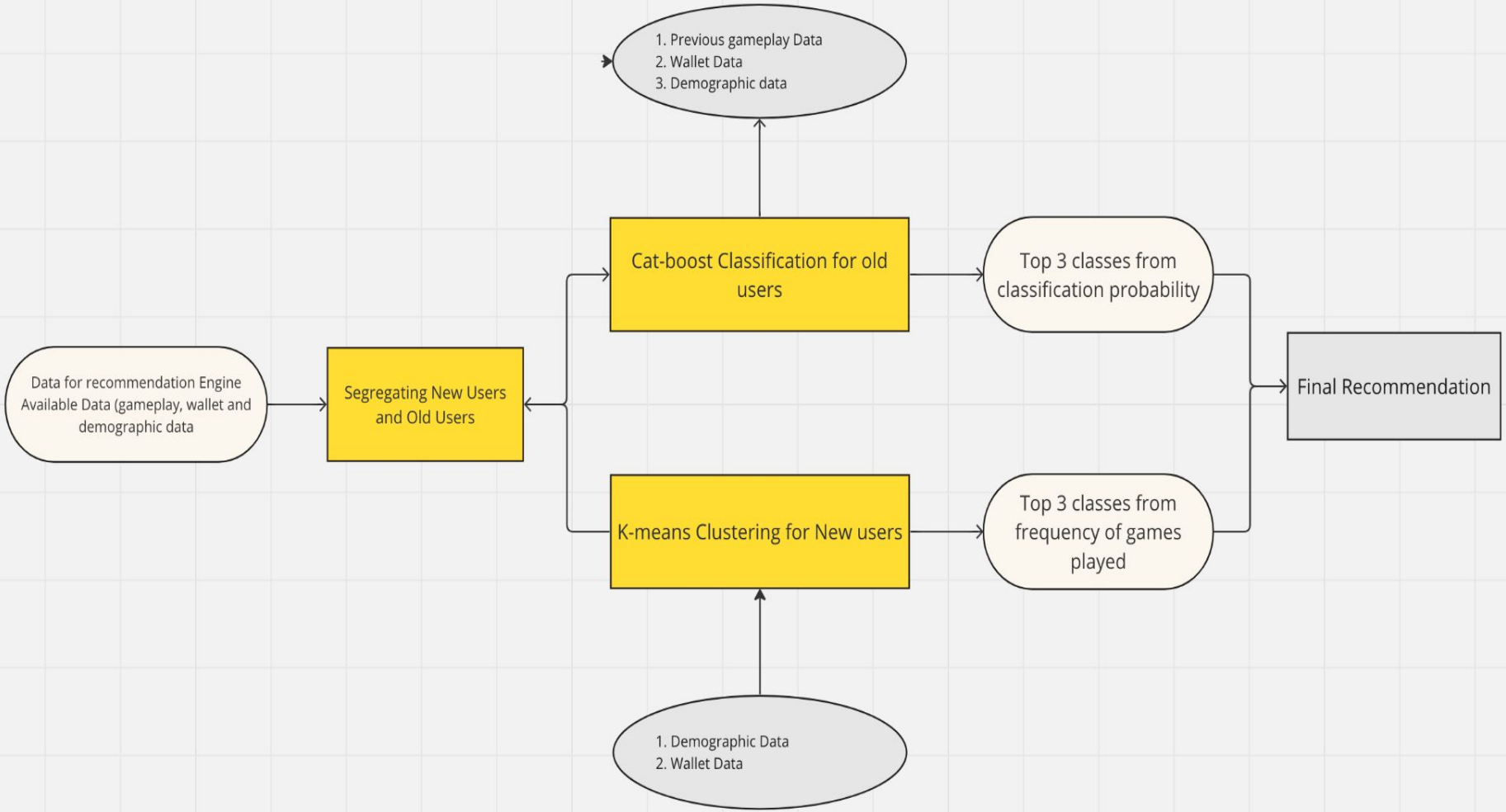
GAME RECOMMENDATION

Rakesh Bobbati

DATA INSIGHTS

- Tournament types are well distinguished with the price points and number of players
- User choices of tournaments and entry fee:
  - Only 46% users stick to single tournament type, 13 % play 2 types and ~9% for each 3 and 4 game types
  - 35% users play more than 3 tournament types out of 4 tournament types
  - 59% play tournament A , D is played by 30% , 9% play C and 2% play B
- Good Correlation in Entry Fee and Number of Games played, at earlier stages people play less fees games
- Good Correlation in Wallet Balance and Number of Games played, at earlier stages people play have less balance
- Users like to play 2 player games, 60% users play 2 player games and 4 player games
- Discrepency in tournament Game Timing. Removed major outliers.
- Users like to play the tournaments they just previously played and won.


1. Previous gameplay Data
2. Wallet Data
3. Demographic data
Cat-boost Classification for old users
Top 3 classes from classification probability
Data for recommendation Engine Available Data (gameplay, wallet and demographic data
Segregating New Users and Old Users
Final Recommendation
K-means Clustering for New users
Top 3 classes from frequency of games played
1. Demographic Data
2. Wallet Data

Recommendation Solution by Seggregation of users

Old User: Idea is to show recommended tournaments based on historical data

This problem fits into a classification problem of Tournament type and combined with payments.

- Used most recent information and historical cumulative data for better recommendation.
- Used recent and cumulative information of tournament, wallet info to create features for the classification models.
- Also used demographic data to create features
- Reason for using Cat-Boost Model
    - Catboost is very good with categorical features as it handles then well.
    - No need to create encodings for categorical data, taken care by the model itself
    - More explainability for predictions with supporting libraries like shapely.
    - Can GPU for faster Training.
    - Tried Random Forest and balanced RF but catboost gave better results

New User: Idea is to increase the adoption, show users which is most engaging

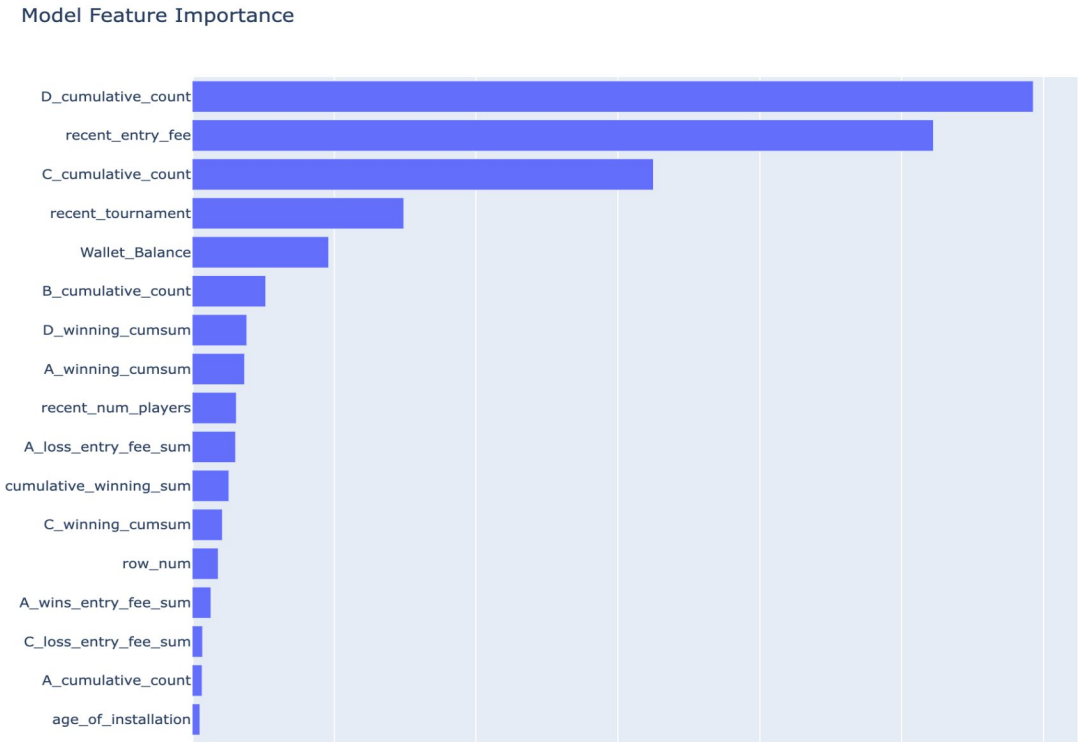
As we donot have much data of the initial user, I used demographicsand wallet info to show in order of games played by similar users.

- Created Clusters based on demographic and device information along with wallet info for some cases.
- Sort the tournaments played in the order frequency, show the tournaments in order of decreasing frequency along with pricing mostly users tend to play low entry fee games and in tournament A i.e – A->D->C->B

Features Engineering for Old Users

- Used Gameplay data along with Wallet balance and Demographic details which includes device details
  - Tournament wise wins and losses cumulative entry fee amount and their respective counts
  - Total amount spent cumulative game by game
  - Frequency of tournamants played cumulative. Total Games played count
  - App installation Age
  - Most recent Entry Fee.
  - Most recent Tournament type
  - Wallet Balance just before playing the Game
  - Game timing and number of players in previous game.

Feature Importance Scores from Model and output




```
clf_catboost.predict_proba(X_test.iloc[:1,:])
```

```
array([[9.98735294e-01, 5.63155595e-04, 7.14578588e-07, 7.35569542e-08,
        2.81416623e-06, 9.49195024e-08, 1.80814899e-06, 1.45300650e-06,
        1.50785057e-07, 1.24048577e-07, 7.48593699e-05, 5.41419139e-05,
        2.88059176e-07, 5.06609452e-06, 3.14908678e-06, 8.42708962e-07,
        1.35734899e-05, 5.42396308e-04]])
```

```
clf_catboost.classes_
```

```
array(['A_1.0', 'A_10.0', 'A_100.0', 'A_1000.0', 'A_25.0', 'A_250.0',
       'A_35.0', 'A_50.0', 'A_500.0', 'A_5000.0', 'B_10.0', 'C_10.0',
       'C_100.0', 'C_25.0', 'C_35.0', 'C_50.0', 'D_15.0', 'D_5.0'],
      dtype=object)
```

```
class_prob = np.argsort(clf_catboost.predict_proba(X_test.iloc[:1,:])[0])[::-1][:3]
```

```
clf_catboost.classes_[class_prob]
```

```
array(['A_1.0', 'A_10.0', 'D_5.0'], dtype=object)
```

Accuracy Of the Model

	precision	recall	f1-score	support
A_1.0	0.99	1.00	1.00	13746
A_10.0	0.99	0.98	0.98	4948
A_100.0	0.95	0.93	0.94	824
A_1000.0	0.96	0.81	0.88	53
A_25.0	0.91	0.94	0.92	2087
A_250.0	0.96	0.92	0.94	285
A_35.0	0.64	0.46	0.54	179
A_50.0	0.93	0.89	0.91	885
A_500.0	0.97	0.84	0.90	104
A_5000.0	1.00	0.50	0.67	4
B_10.0	0.99	1.00	1.00	779
C_10.0	0.93	1.00	0.96	1973
C_100.0	0.96	0.89	0.92	475
C_25.0	0.60	0.14	0.22	111
C_35.0	0.84	0.90	0.87	542
C_50.0	0.84	0.78	0.81	291
D_15.0	0.94	0.81	0.87	442
D_5.0	0.98	1.00	0.99	3396
accuracy			0.97	31124
macro avg	0.91	0.82	0.85	31124
weighted avg	0.97	0.97	0.97	31124

Features Engineering for New Users

- Used Demographic details which includes device details and Wallet Info
  - Demographic details for location , device and connection are frequency encoded along with wallet balance
  - Kmeans Clustering is used, hyperparameter tuned with elbow method for WCSS
  - WCSS is the sum of the squared distance between each point and the centroid in a cluster.
  - Most Frequently played games are recommended for the respective clusters.

Game_category	A_1.0	A_10.0	A_100.0	A_25.0	A_250.0	A_35.0	A_50.0	B_10.0	C_10.0	C_100.0	C_25.0	C_35.0	D_15.0	D_5.0
0	152.0	18.0	1.0	NaN	NaN	17.0	1.0	NaN	4.0	NaN	2.0	NaN	NaN	13.0
1	1511.0	166.0	NaN	3.0	NaN	160.0	4.0	4.0	37.0	NaN	39.0	1.0	NaN	81.0
2	113.0	25.0	NaN	NaN	NaN	25.0	NaN	NaN	7.0	1.0	NaN	NaN	NaN	5.0
3	398.0	55.0	NaN	1.0	NaN	47.0	1.0	NaN	11.0	NaN	4.0	NaN	NaN	43.0
4	749.0	60.0	4.0	3.0	1.0	31.0	NaN	2.0	11.0	NaN	3.0	NaN	NaN	32.0
5	195.0	10.0	NaN	NaN	NaN	1.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	6.0
6	340.0	42.0	NaN	NaN	NaN	25.0	1.0	1.0	8.0	NaN	6.0	NaN	NaN	24.0
7	75.0	24.0	NaN	NaN	NaN	15.0	NaN	NaN	7.0	NaN	1.0	NaN	NaN	4.0
8	1018.0	132.0	3.0	2.0	NaN	47.0	3.0	1.0	28.0	1.0	19.0	NaN	1.0	72.0
9	299.0	39.0	NaN	NaN	NaN	16.0	2.0	1.0	8.0	NaN	1.0	1.0	1.0	27.0

Improvements Scope

- Hyperparameter Tuning of the models and better balancing the dataset.
- When New tournaments come , we need to retrain the models. Also we can try collaborative filtering.

Thank You