Team Max

Team 4 – Projekt 3 Churn Prediction

Agenda

- Introduction
- Approaches
- Implementation
- Experiments
- Results
- Conclusion

Introduction

What is churn prediction?

Churn prediction is predicting which customers or players are at high risk of leaving your product or canceling a subscription to a service, based on their behavior with your product.

Introduction

The Dataset

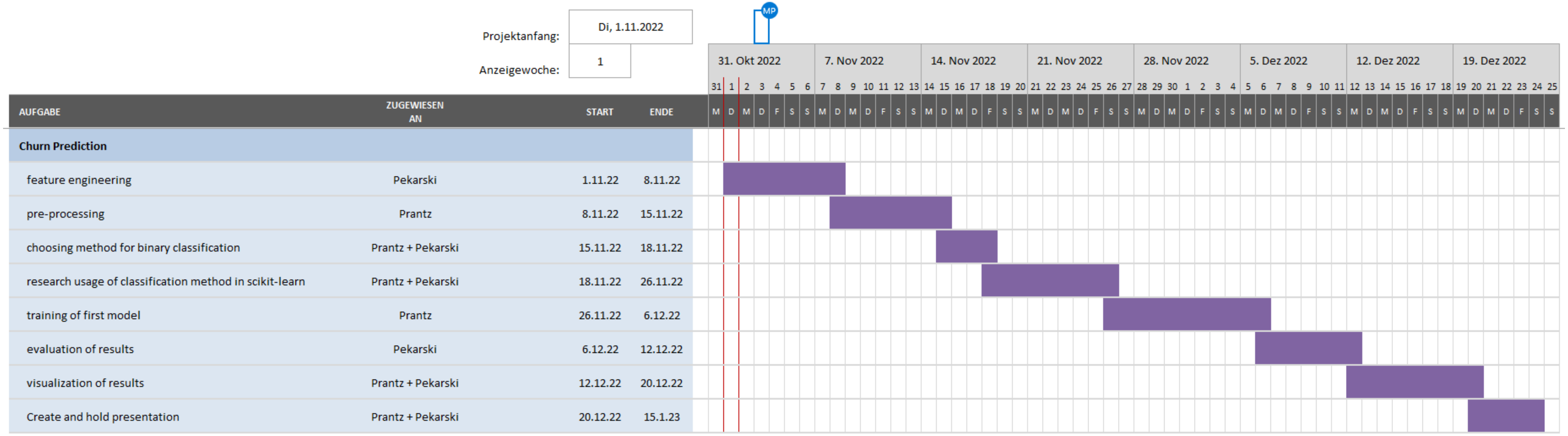
- 1 Train set
  - 4000 Players
  - 44gb of Data
  - 74 Columns

The Goal:

- Predict which players leave, who will stay
- Be better than y = 1 (F1 ± 0.46)
- Learning binary classification
- Figure out best features & methods for this use case

Introduction

Ursprünglicher Zeitplan



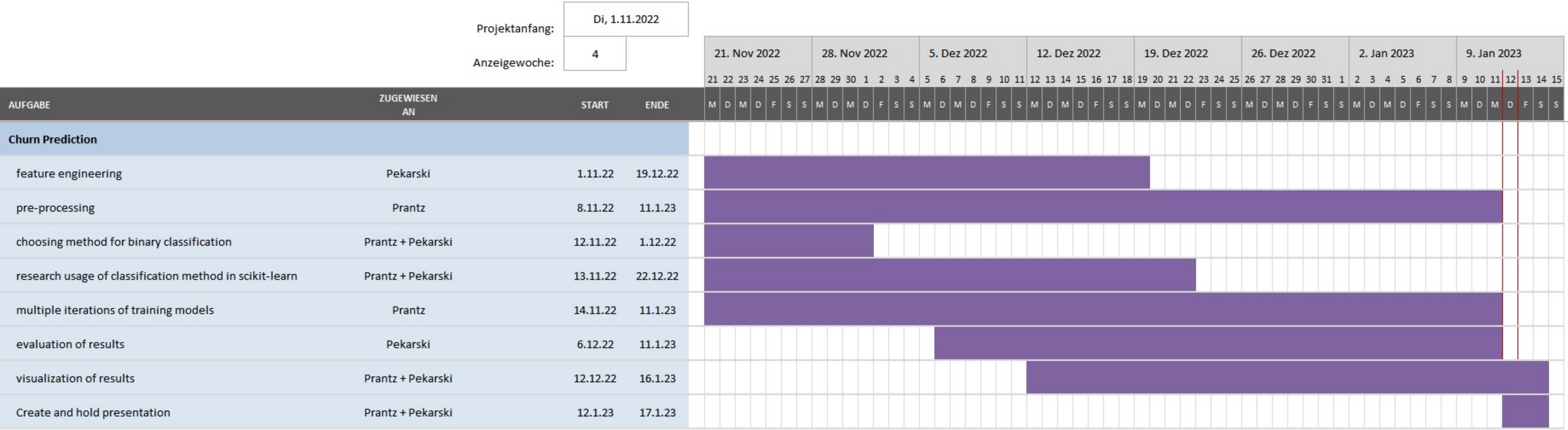
Projektanfang: Di, 1.11.2022

Anzeigewoche: 1

AUFGABE	ZUGEWIESEN AN	START	ENDE
Churn Prediction			
feature engineering	Pekarski	1.11.22	8.11.22
pre-processing	Prantz	8.11.22	15.11.22
choosing method for binary classification	Prantz + Pekarski	15.11.22	18.11.22
research usage of classification method in scikit-learn	Prantz + Pekarski	18.11.22	26.11.22
training of first model	Prantz	26.11.22	6.12.22
evaluation of results	Pekarski	6.12.22	12.12.22
visualization of results	Prantz + Pekarski	12.12.22	20.12.22
Create and hold presentation	Prantz + Pekarski	20.12.22	15.1.23

Introduction

Tatsächlicher Zeitplan



Projektanfang: Di, 1.11.2022

Anzeigewoche: 4

AUFGABE	ZUGEWIESEN AN	START	ENDE
Churn Prediction			
feature engineering	Pekarski	1.11.22	19.12.22
pre-processing	Prantz	8.11.22	11.1.23
choosing method for binary classification	Prantz + Pekarski	12.11.22	1.12.22
research usage of classification method in scikit-learn	Prantz + Pekarski	13.11.22	22.12.22
multiple iterations of training models	Prantz	14.11.22	11.1.23
evaluation of results	Pekarski	6.12.22	11.1.23
visualization of results	Prantz + Pekarski	12.12.22	16.1.23
Create and hold presentation	Prantz + Pekarski	12.1.23	17.1.23

Approaches

- Data exploration
- Data cleaning
- Feature engineering
- Classification models?
  - Decision tree
  - Random Forrest
  - Neural Net
  - Gaussian Process Classifier

Implementation

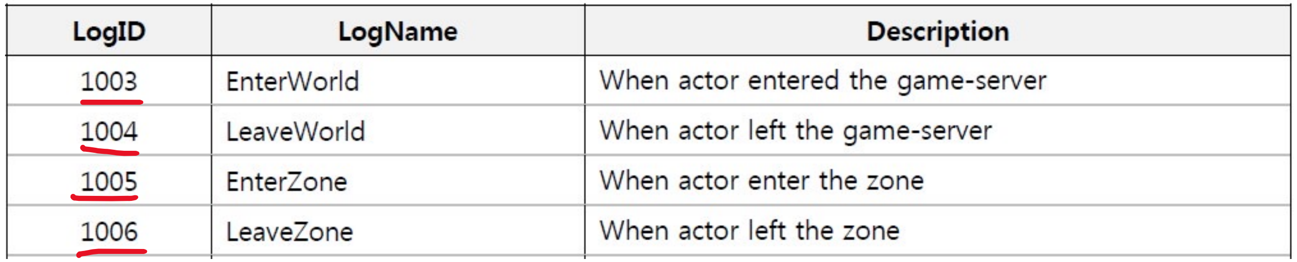
- Features
  - Available attributes
  - Importance of attributes

Seq
TIME
LogID
Session_ID
Link_ID
Log_Detail_Code
Actor_Code
Actor_ID
Actor_Account_ID
Actor_Object_ID
Actor_Zone_ID
Actor_Zone_UID
Actor_Zone_Channel_ID
Actor_Party_ID
Actor_Team_ID
Actor_Option1_NUM
Actor_Server
Actor_Guild
Actor_Level
Actor_Race
Actor_Job
Actor_Faction
Actor_Faction2
Actor_MasteryLevel
Actor_Gender
Actor_Option2_STR

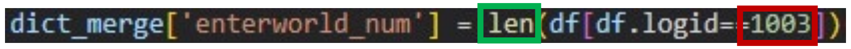
Old_Value1_STR
Old_Value2_NUM
Old_Value3_NUM
Old_Value4_NUM
Use_Value1_NUM
Use_Value2_NUM
Use_Value3_NUM
New_Value1_STR
New_Value2_NUM
New_Value3_NUM
New_Value4_NUM
Data1_NUM
Data2_NUM
Data3_NUM
Data4_NUM
Data5_NUM
Data6_NUM
Data7_NUM

Implementation

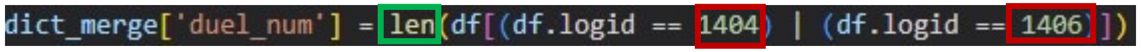
- Features
  - Types of aggregation
  - iterative process



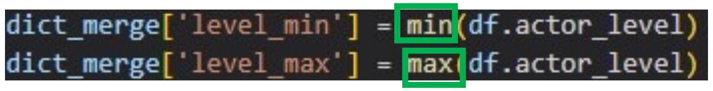
LogID	LogName	Description
1003	EnterWorld	When actor entered the game-server
1004	LeaveWorld	When actor left the game-server
1005	EnterZone	When actor enter the zone
1006	LeaveZone	When actor left the zone



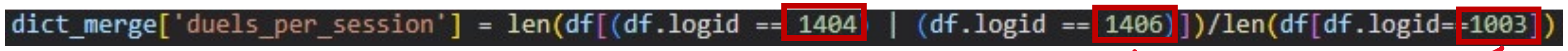
```
dict_merge['enterworld_num'] = len(df[df.logid==1003])
```



```
dict_merge['duel_num'] = len(df[(df.logid == 1404) | (df.logid == 1406)])
```



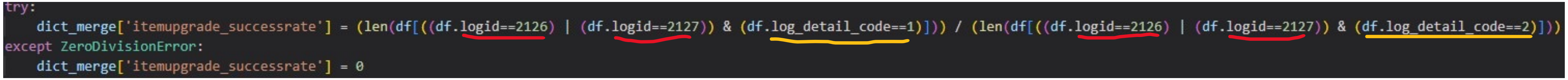
```
dict_merge['level_min'] = min(df.actor_level)
dict_merge['level_max'] = max(df.actor_level)
```



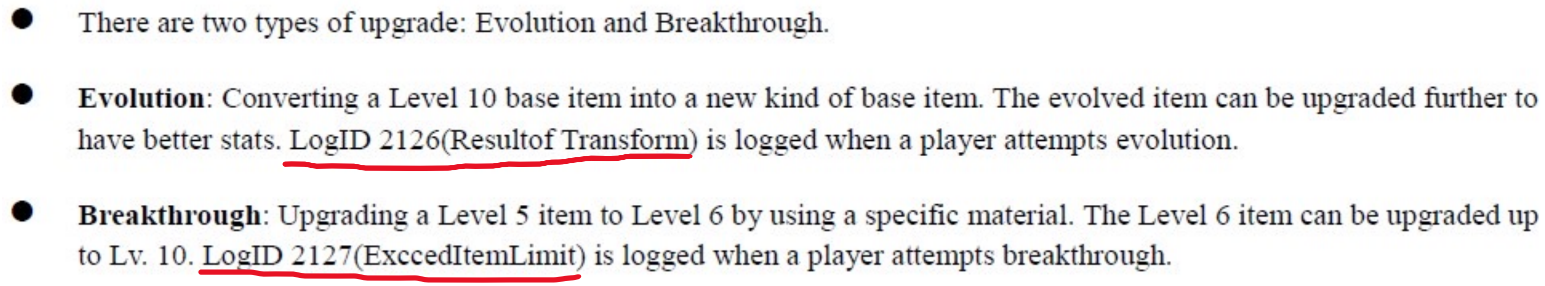
```
dict_merge['duels_per_session'] = len(df[(df.logid == 1404) | (df.logid == 1406)])/len(df[df.logid==1003])
```

len(1v1 Duel OR TeamDuel) / len (logins)

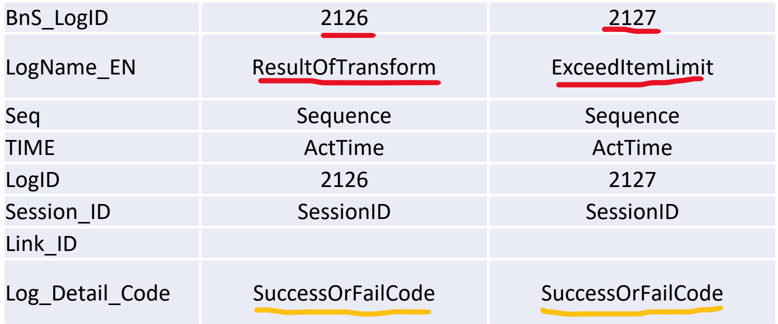
Implementation



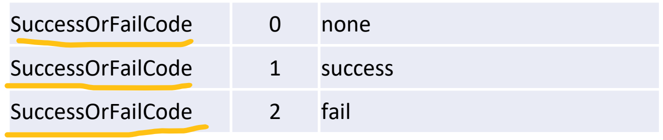
```
try:
    dict_merge['itemupgrade_successrate'] = (len(df[((df.logid==2126) | (df.logid==2127)) & (df.log_detail_code==1)])) / (len(df[((df.logid==2126) | (df.logid==2127)) & (df.log_detail_code==2)]))
except ZeroDivisionError:
    dict_merge['itemupgrade_successrate'] = 0
```



- There are two types of upgrade: Evolution and Breakthrough.
- **Evolution**: Converting a Level 10 base item into a new kind of base item. The evolved item can be upgraded further to have better stats. LogID 2126(Resultof Transform) is logged when a player attempts evolution.
- **Breakthrough**: Upgrading a Level 5 item to Level 6 by using a specific material. The Level 6 item can be upgraded up to Lv. 10. LogID 2127(ExccedItemLimit) is logged when a player attempts breakthrough.

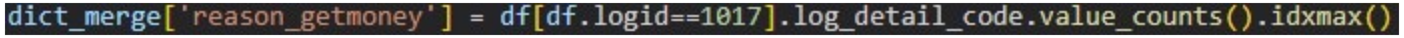


BnS_LogID	2126	2127
LogName_EN	ResultOfTransform	ExceedItemLimit
Seq	Sequence	Sequence
TIME	ActTime	ActTime
LogID	2126	2127
Session_ID	SessionID	SessionID
Link_ID		
Log_Detail_Code	SuccessOrFailCode	SuccessOrFailCode



SuccessOrFailCode	0	none
SuccessOrFailCode	1	success
SuccessOrFailCode	2	fail

Implementation



```
dict_merge['reason_getmoney'] = df[df.logid==1017].log_detail_code.value_counts().idxmax()
```

BnS_LogID	1017
LogName_EN	GetMoney
Seq	Sequence
TIME	ActTime
LogID	1017
Session_ID	SessionID
Link_ID	
Log_Detail_Code	GetMoneyReasonCode

GetMoneyReasonCode	100	None
GetMoneyReasonCode	101	GetLootMoney
GetMoneyReasonCode	102	GetMoneyPostUnknown
GetMoneyReasonCode	103	GetMoneyPostFromUser
GetMoneyReasonCode	104	GetMoneyPostFromTool
GetMoneyReasonCode	105	GetMoneyPostGathering
GetMoneyReasonCode	106	GetMoneyPostProduction
GetMoneyReasonCode	107	GetMoneyPostSaleSuccess
GetMoneyReasonCode	108	GetMoneyPostBidLoser
GetMoneyReasonCode	109	GetMoneyPostBidAbandonment
GetMoneyReasonCode	110	ReceiveExpressPostFail
GetMoneyReasonCode	111	ProductionFail
GetMoneyReasonCode	112	CreateGuildFail
GetMoneyReasonCode	114	DepositMoneyInGuildBankFail
GetMoneyReasonCode	116	DebugCommandSetMoney
GetMoneyReasonCode	146	ChangeWeaponAppearanceFail
GetMoneyReasonCode	151	TradeGetMoney
GetMoneyReasonCode	152	SellItem
GetMoneyReasonCode	161	QuestReward
GetMoneyReasonCode	165	CompleteChallengeToday
GetMoneyReasonCode	168	DistributePartyAuctionMiscarriedMoney
GetMoneyReasonCode	169	DistributePartyAuctionSoldMoney
GetMoneyReasonCode	171	PutMainAuctionFail
GetMoneyReasonCode	172	BidMainAuctionFail
GetMoneyReasonCode	173	RebidMarketSaleFail
GetMoneyReasonCode	174	BuyItemNowMainAuctionFail
GetMoneyReasonCode	176	take-rollback-by-fail-refresh-simple-quest-pack

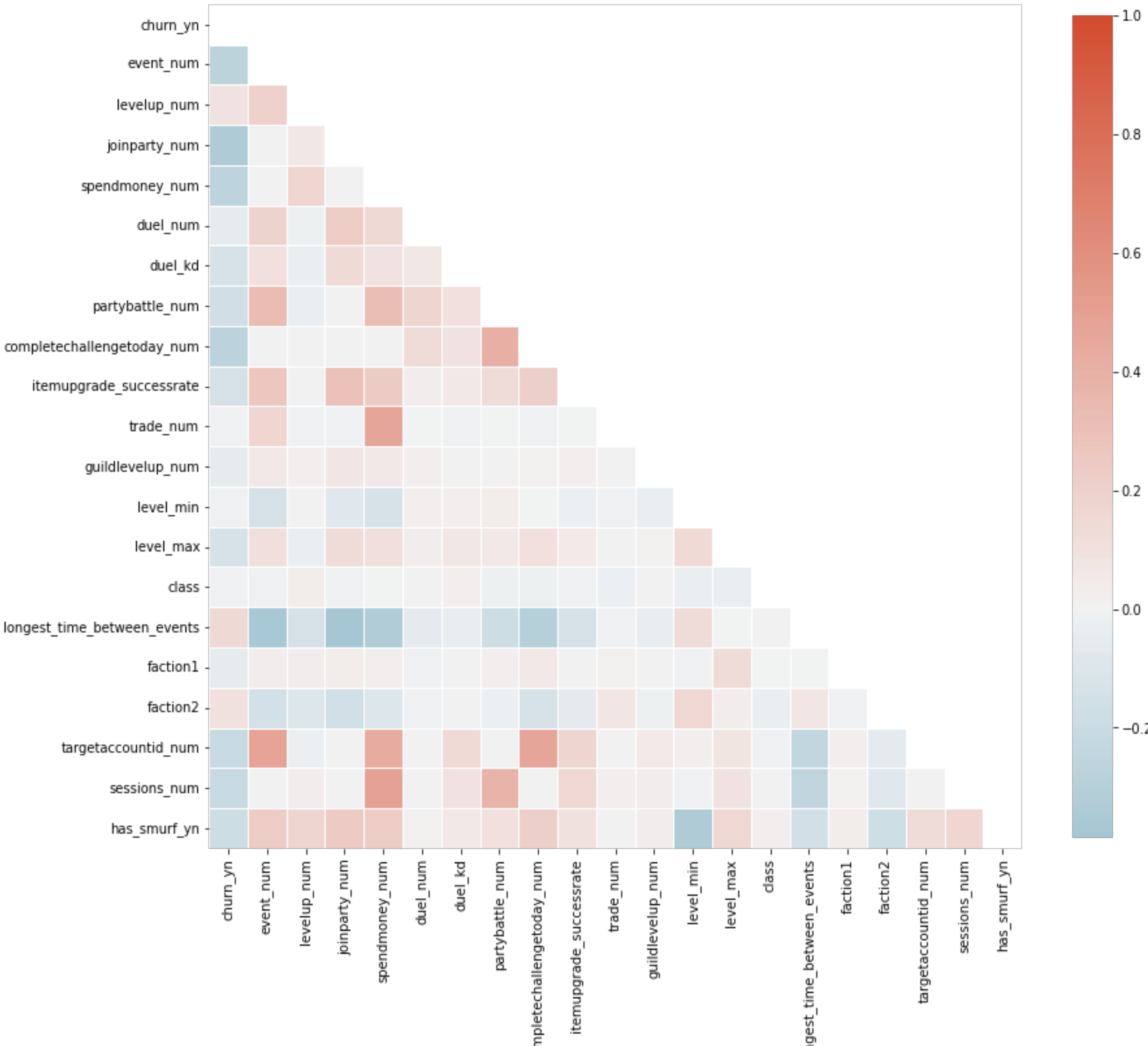
Implementation

- 38 features
- 30 attributes used
- 9 types of aggregation

actor_account_id
churn_yn
survival_time
event_num
enterworld_num
levelup_num
joinparty_num
spendmoney_num
average_money_spent_per_session
duel_num
duel_kd
partybattle_num
completechallengetoday_num
completechallengeweek_num
itemupgrade_successrate
trade_num
buyitemnowmainauction_num
guildlevelup_num
level_min
level_max
class
longest_time_between_events
average_time_between_events
average_time_between_logins
faction1
faction2
targetaccountid_num
sessions_num
masteryexp
duelpoints_max
partybattlepoints_max
duel_rating_score_max
money_max
gathering_num
has_smurf_yn
duels_per_session
reason_getmoney
reason_spendmoney

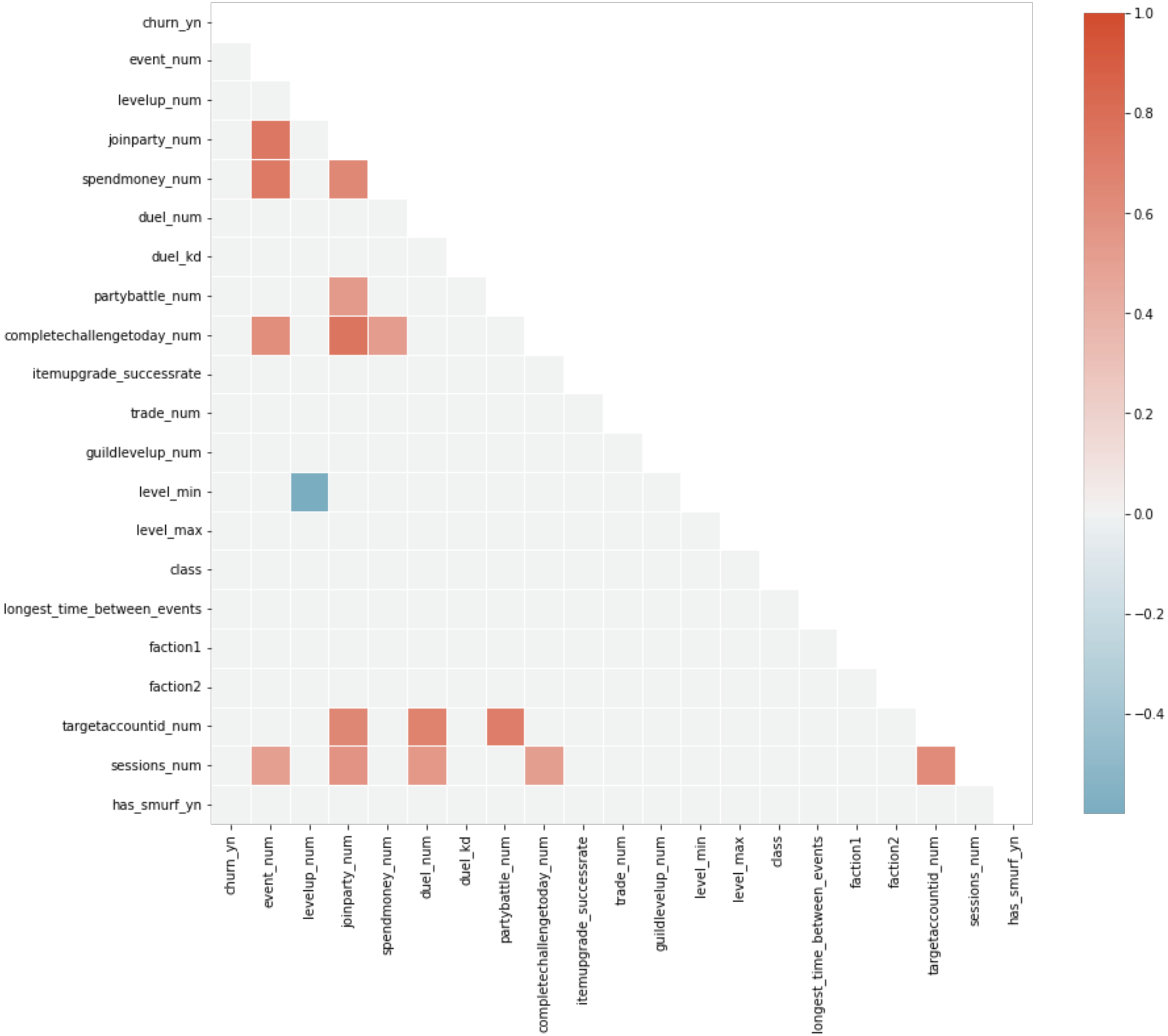
Experiments

- Data insight



Experiments

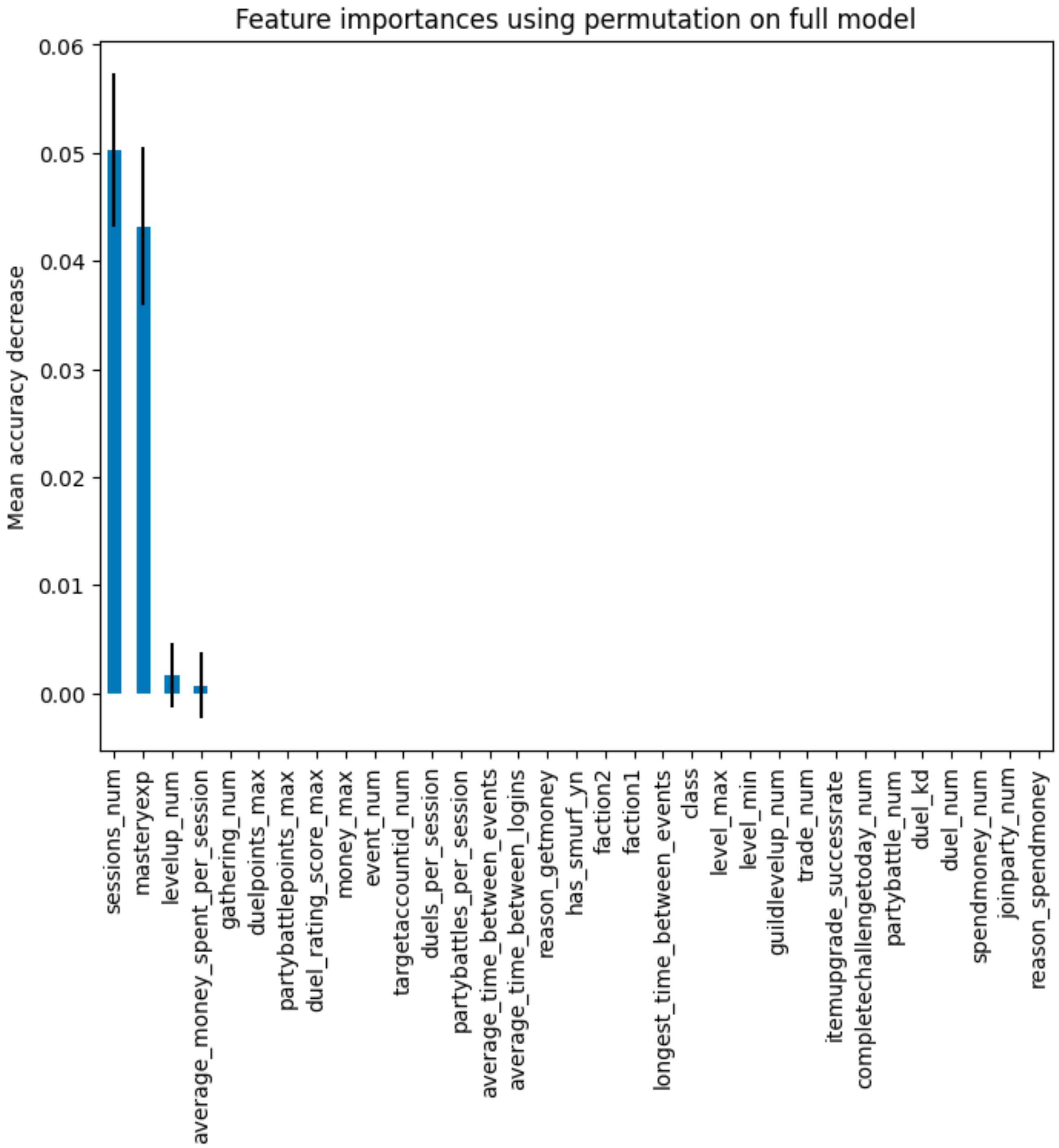
- Data insight
  - Correlation <50%



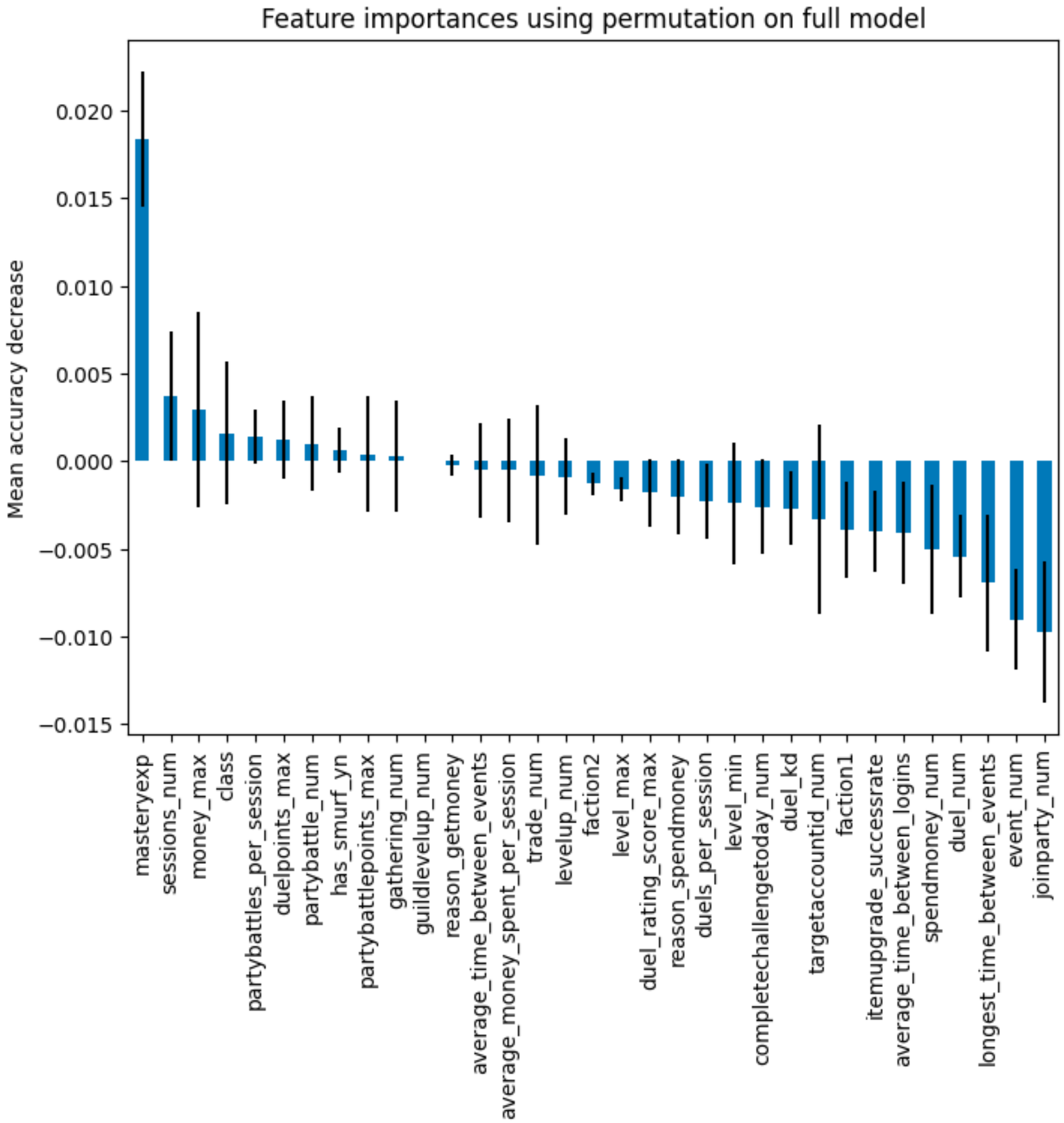
Experiments

- Models
  - Decision Tree
  - Random Forrest
  - Neural Net
  - Gaussian Process Classifier
  - Voting Classifier
  - XGB
  - Others

Decision Tree



Random Forrest

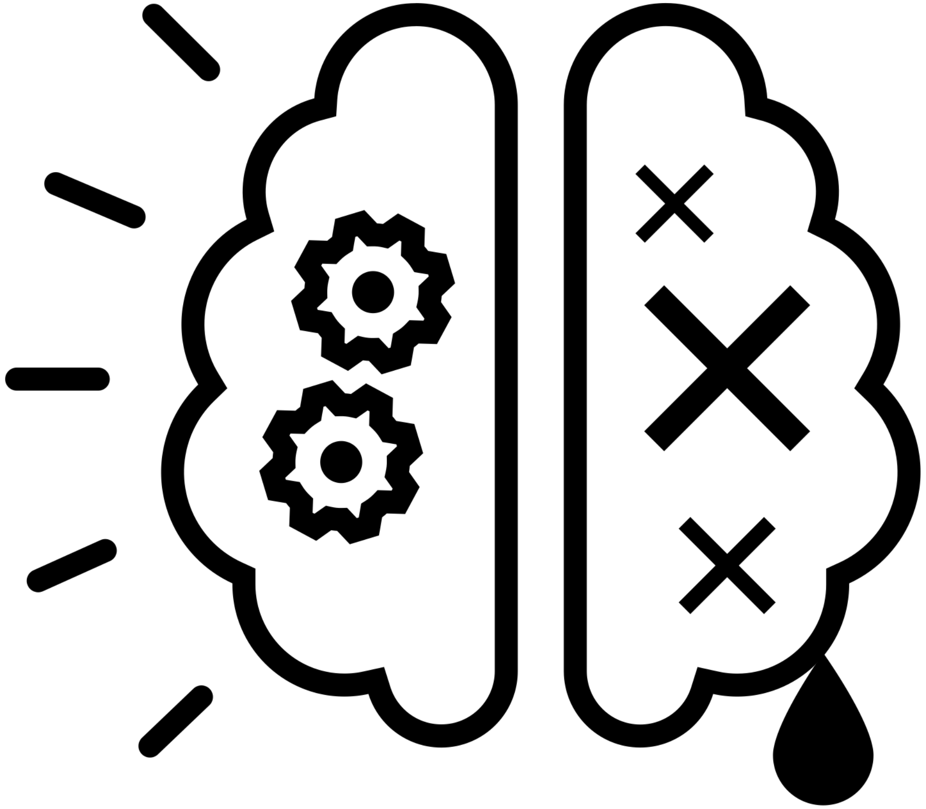


Neural net

- Abysmal accuracy & F1

Gaussian Process Classifier

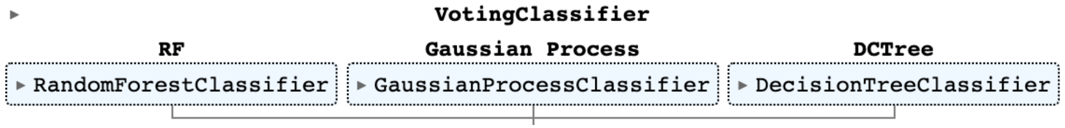
- Can’t even give back feature importance
- But good scores, at least :)



Voting Classifier

F1:

0.55 with a standard deviation of 0.03



▸ VotingClassifier

RF ▸ RandomForestClassifier

Gaussian Process ▸ GaussianProcessClassifier

DCTree ▸ DecisionTreeClassifier

Why voting Classifier?

- ”hive mind”
- Might find players a single classification couldn’t have found
- The more good classifiers are combined, the better might be the result. Diversity is important!
- No need for the “single perfect” classifier
- The better the single classifiers, the better the result
- So, why not?
  - (Takes a bit more time to fine tune, IF it’s fine tunable at all)
  - Dependent on used classifiers, can’t print or weight features as good

Conclusion

- The more features the merrier
- Good documentation is priceless
- Every step turned out to be an iterative process
- Neural nets tend to be really bad with this Dataset.
- Normalization is important (1% performance uplfit)
  - SK standard scaler fitted to X_Train x_test
- Tree & Random forrest come with a relatively good result out of the box
- Same goes for Gaussian process classifier
- After a certain point, time put into feature engineering would have been more efficient
- Was fun ☺

Questions?