I am what I am in MOBA games: A machine learning approach using gaming behaviors to predict employees' personalities

Shubin Yu, Shubin.Yu@bi.no
Anqi Yu

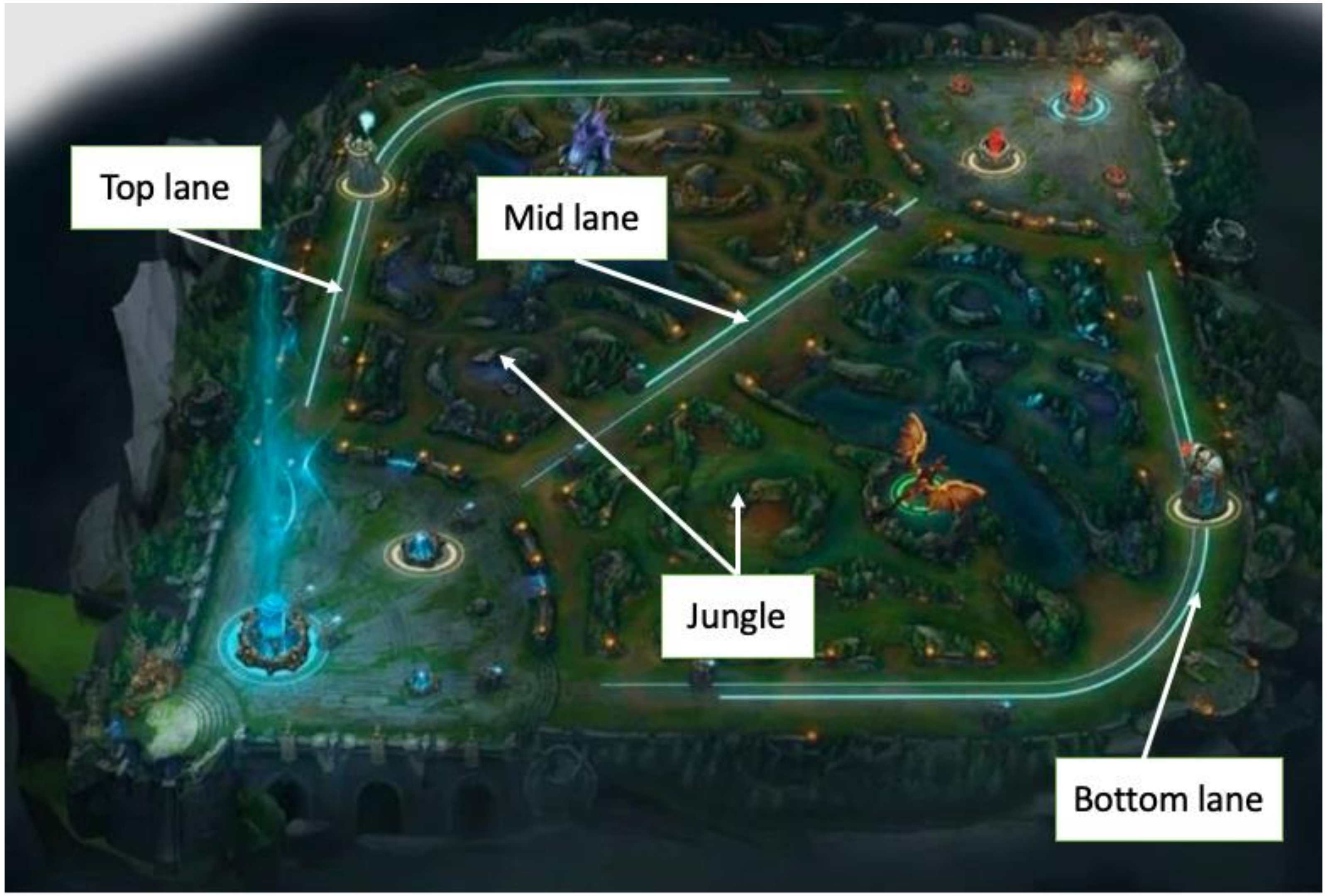
LEAGUE OF LEGENDS

HONOR OF KINGS
Rexxar
364
364
Crazyxx
15
Saiki
15
Immune
15
98
96
Iam
15
PLAY YOUR STYLE
CARRY THE TEAM

The popularity of MOBA games

- Multiplayer online battle arena (MOBA) games are among the most popular game types around the world.
- League of Legends, Honor of Kings, Arena of Valor, Dota 2, Heroes of the Storm
- In 2020, around 100 million Chinese played Honor of Kings every day (techcrunch, 2020).
- League of Legends also recorded monthly active users of around 150 million in 2022 (Activeplayer.io, 2022).
- Employees play MOBA games with each other and even with their managers during lunch breaks or after work.

644



The position in a typical MOBA battlefield

The six types of roles

Tank

Tanks are characters who are incredibly **resilient** and good at **absorbing a lot of damage**. They frequently have skills that allow them to exert control over the battlefield, such as crowd control or area-of-effect damage. Malphite and Leona are two examples of tanks.

Bruisers/fighters

Bruisers: These characters are often a hybrid of tanks and fighters; they are **tough** and capable of dealing significant damage. They frequently have skills that allow them to exert control over the battlefield, such as crowd control or area-of-effect damage. Darius and Garen are two examples of bruisers.

Ability Power Carry (APC)/Mage

APCs (a.k.a Mages) are often extremely powerful spellcasters who excel at dealing massive quantities of damage to several targets at once. They frequently have skills that allow them to **exert control** over the battlefield, such as crowd control or area-of-effect damage. APC examples include Lux and Veigar.

Support

These characters are often focused on **supporting the team** and have skills that allow the squad to deal more damage and live for longer periods of time. They frequently have skills that allow them to exert control over the battlefield, such as crowd control or area-of-effect damage. Soraka and Janna are two examples of support characters.

Assassins

These characters are usually incredibly mobile and adept at dealing massive quantities of damage to a single victim in a short period of time. They frequently have skills that allow them to **quickly move into and out of combat** and can be difficult to catch. Zed and Akali are two examples of assassins.

Attack Damage Carry (ADC)

These characters are mainly centered on doing **large physical damage**, and their powers scale with their attack damage. They frequently feature skills that allow them to influence the battlefield, such as crowd control effects or area-of-effect damage. Vayne and Varus are two examples of AD Carry.

Research questions

This study seeks to determine whether it is possible to predict one's personality based on gaming behaviors, namely the choice of position, role, and skill selection.

In particular, we would like to understand:

Which personality can be predicted?

Methods

Data collection:

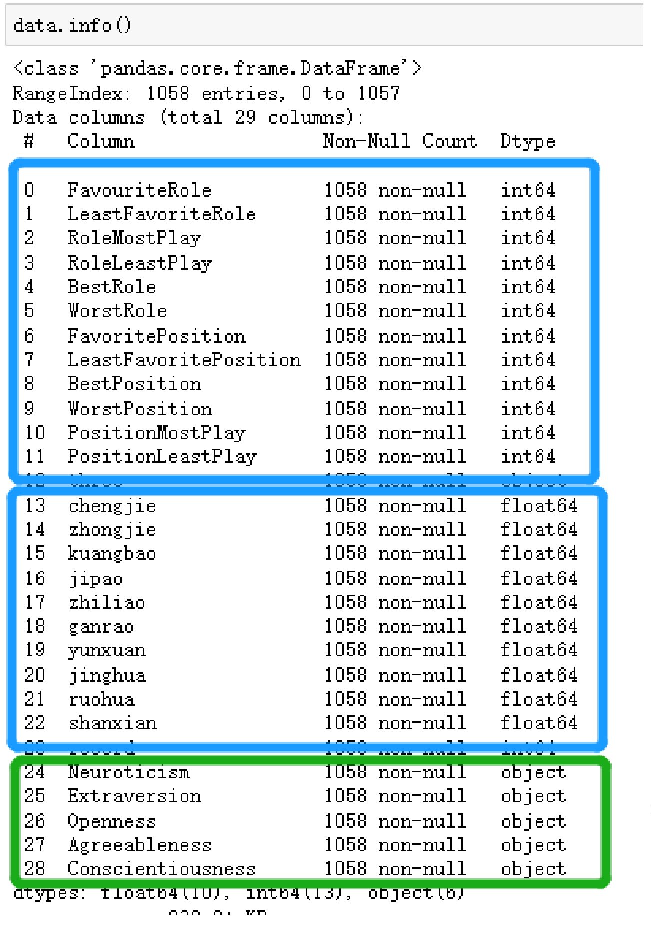
A survey study, 1,058 respondents, Credamo

Participants: People who have played the game "Honor of Kings" and who are currently employees.
Measures: Most/least favorite position, position that they are best/worst in, most/least frequently position played, most/least favorite role type, role type that they are best/worst in, most/least frequently role type played, and most favorite skills to use

NEO Five-Factor Inventory (NEO-FFI), a 60-item self-report questionnaire that measures the five dimensions of personality.

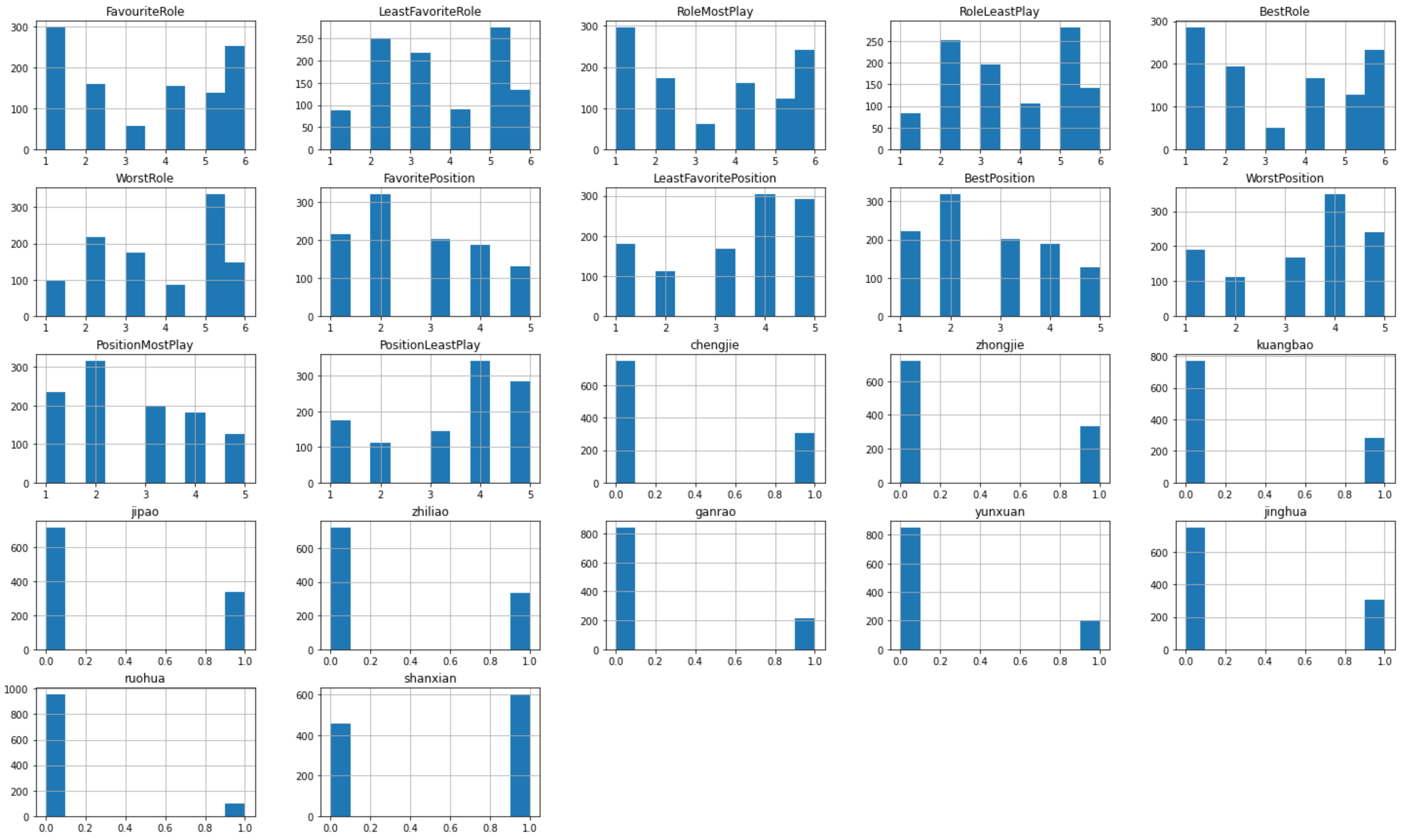
Dimension	Items	Cronbach's alpha
Neuroticism	Q1 Q6 Q11 Q16 Q21 Q26 Q31 Q36 Q41 Q46 Q51 Q56	.92
Extraversion	Q2 Q7 Q12 Q17 Q22 Q27 Q32 Q37 Q42 Q47 Q52 Q57	.83
Openness	Q3 Q8 Q13 Q18 Q23 Q28 Q33 Q38 Q43 Q48 Q53 Q58	.79
Agreeableness	Q4 Q9 Q14 Q19 Q24 Q29 Q34 Q39 Q44 Q49 Q54 Q59	.73
Conscientiousness	Q5 Q10 Q15 Q20 Q25 Q30 Q35 Q40 Q45 Q50 Q55 Q60	.86

Data exploration

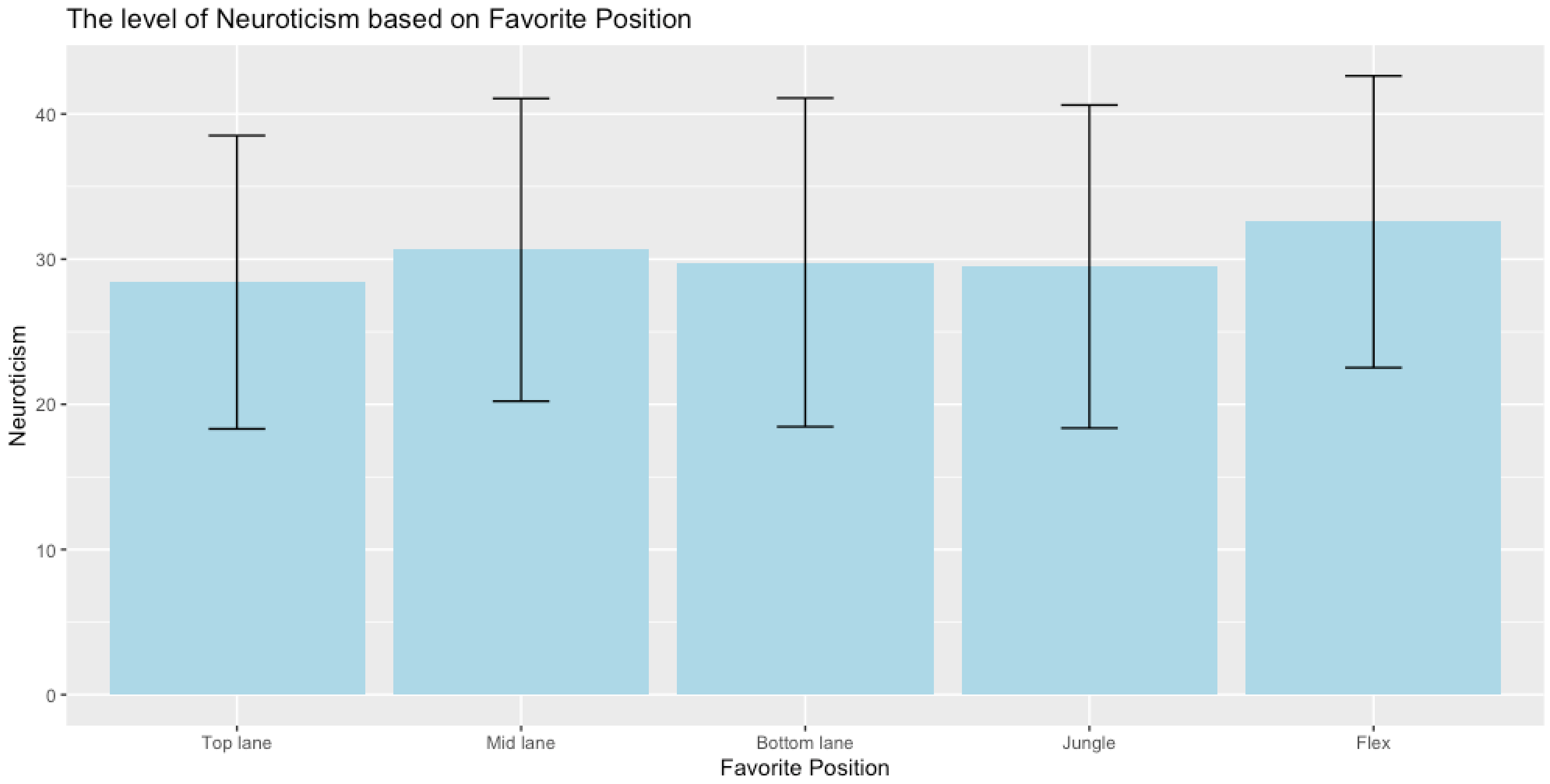


```
data.info()
```

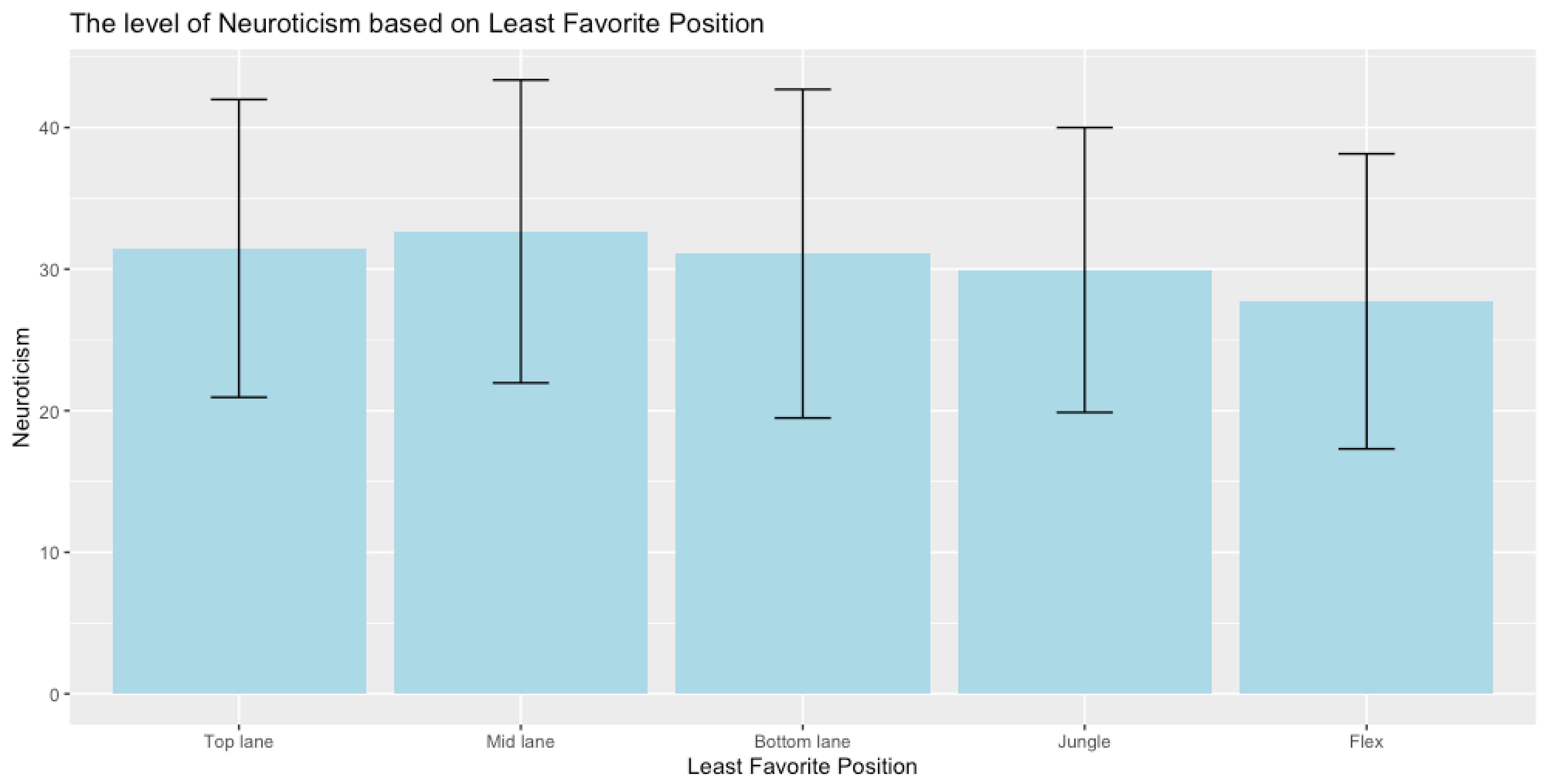
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1058 entries, 0 to 1057
Data columns (total 29 columns):
 #   Column                Non-Null Count  Dtype
 0   FavouriteRole         1058 non-null   int64
 1   LeastFavoriteRole     1058 non-null   int64
 2   RoleMostPlay          1058 non-null   int64
 3   RoleLeastPlay         1058 non-null   int64
 4   BestRole              1058 non-null   int64
 5   WorstRole             1058 non-null   int64
 6   FavoritePosition      1058 non-null   int64
 7   LeastFavoritePosition 1058 non-null   int64
 8   BestPosition          1058 non-null   int64
 9   WorstPosition         1058 non-null   int64
 10  PositionMostPlay      1058 non-null   int64
 11  PositionLeastPlay     1058 non-null   int64
 12  three                 1058 non-null   object
 13  chengjie              1058 non-null   float64
 14  zhongjie              1058 non-null   float64
 15  kuangbao              1058 non-null   float64
 16  jipao                 1058 non-null   float64
 17  zhiliao               1058 non-null   float64
 18  ganrao                1058 non-null   float64
 19  yunxuan               1058 non-null   float64
 20  jinghua               1058 non-null   float64
 21  ruohua                1058 non-null   float64
 22  shanxian              1058 non-null   float64
 23  record                1058 non-null   int64
 24  Neuroticism           1058 non-null   object
 25  Extraversion          1058 non-null   object
 26  Openness              1058 non-null   object
 27  Agreeableness         1058 non-null   object
 28  Conscientiousness     1058 non-null   object
dtypes: float64(10), int64(13), object(6)
[illegible]
```



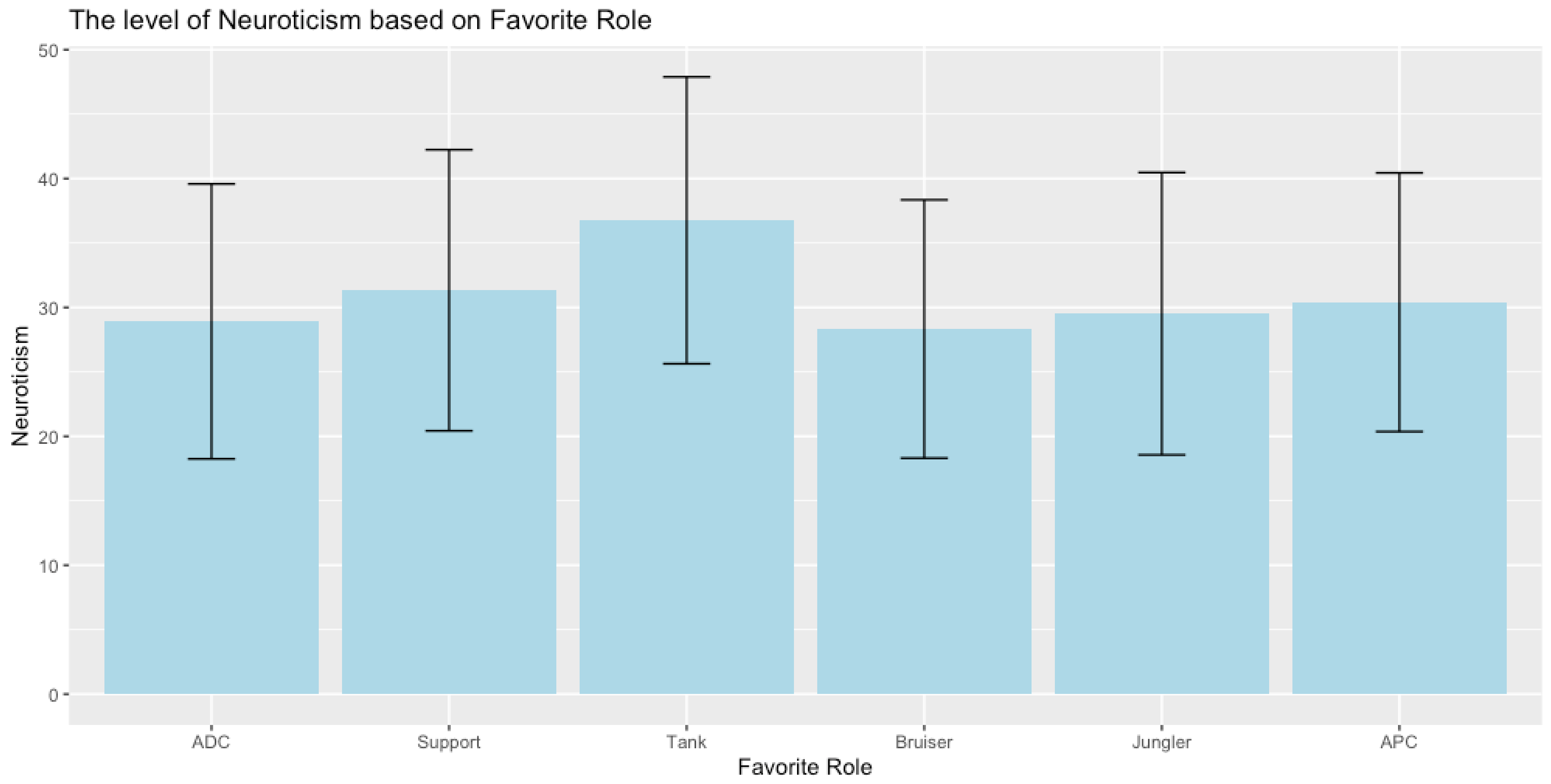
Data exploration



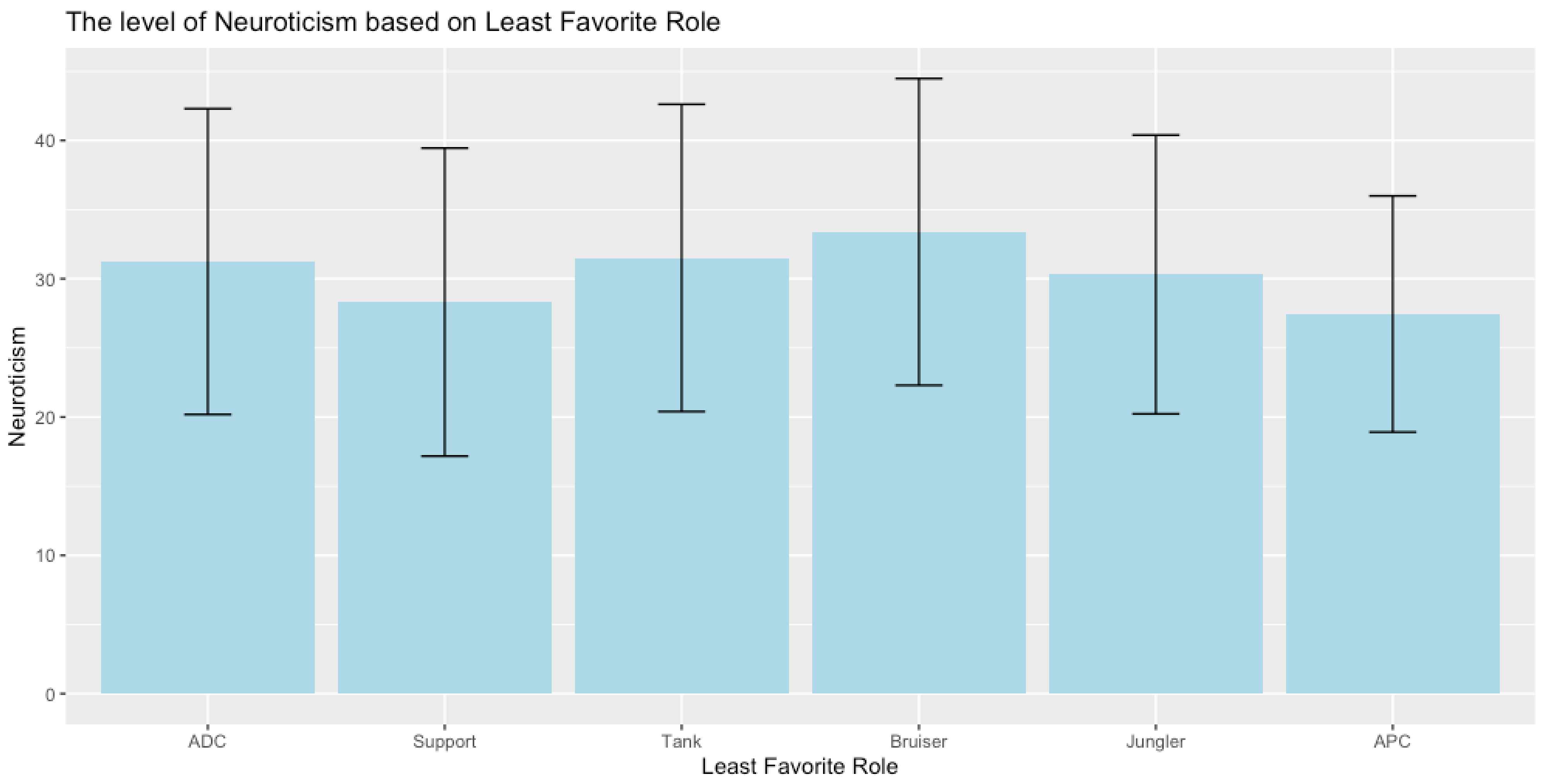
Data exploration



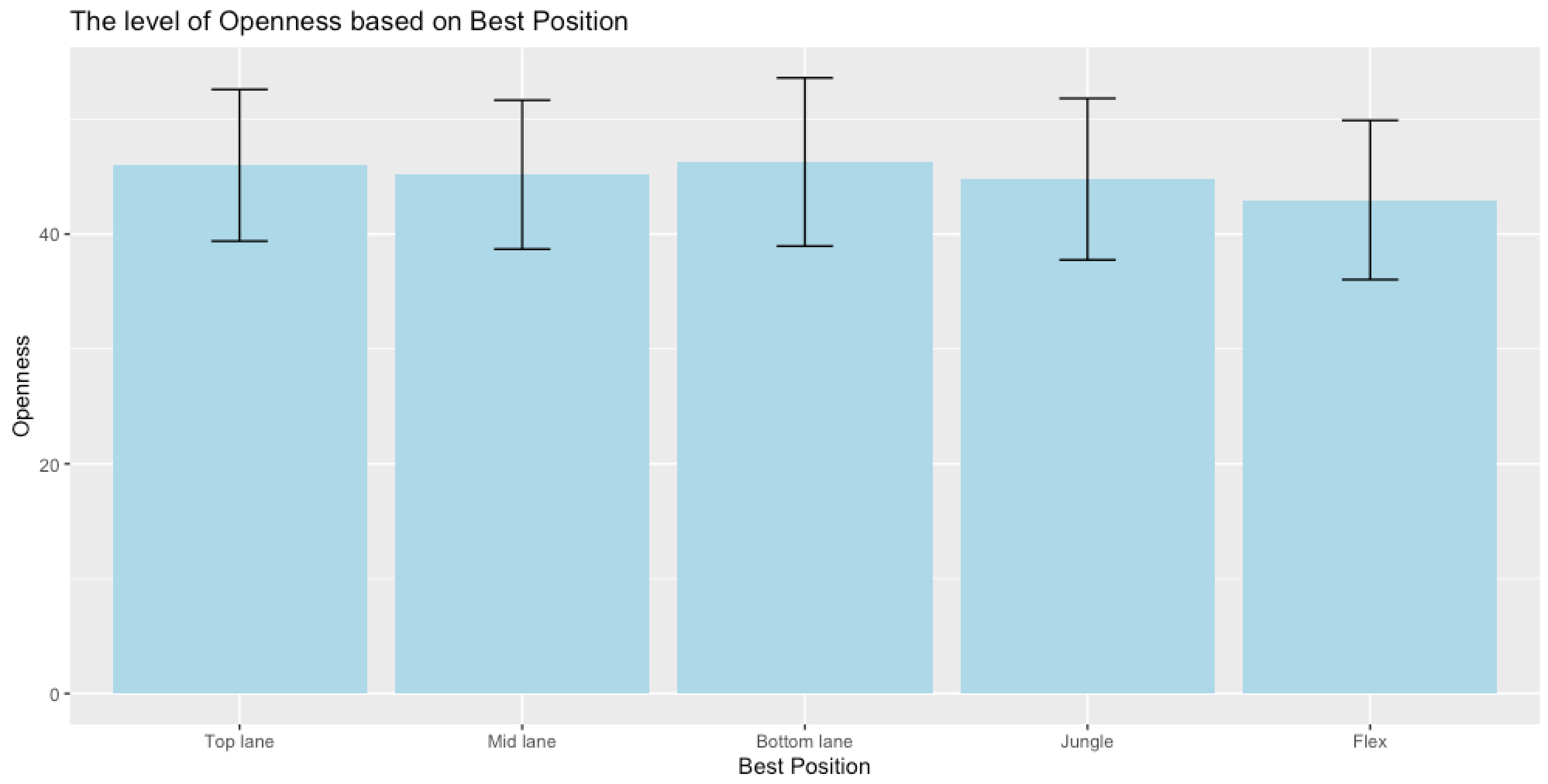
Data exploration



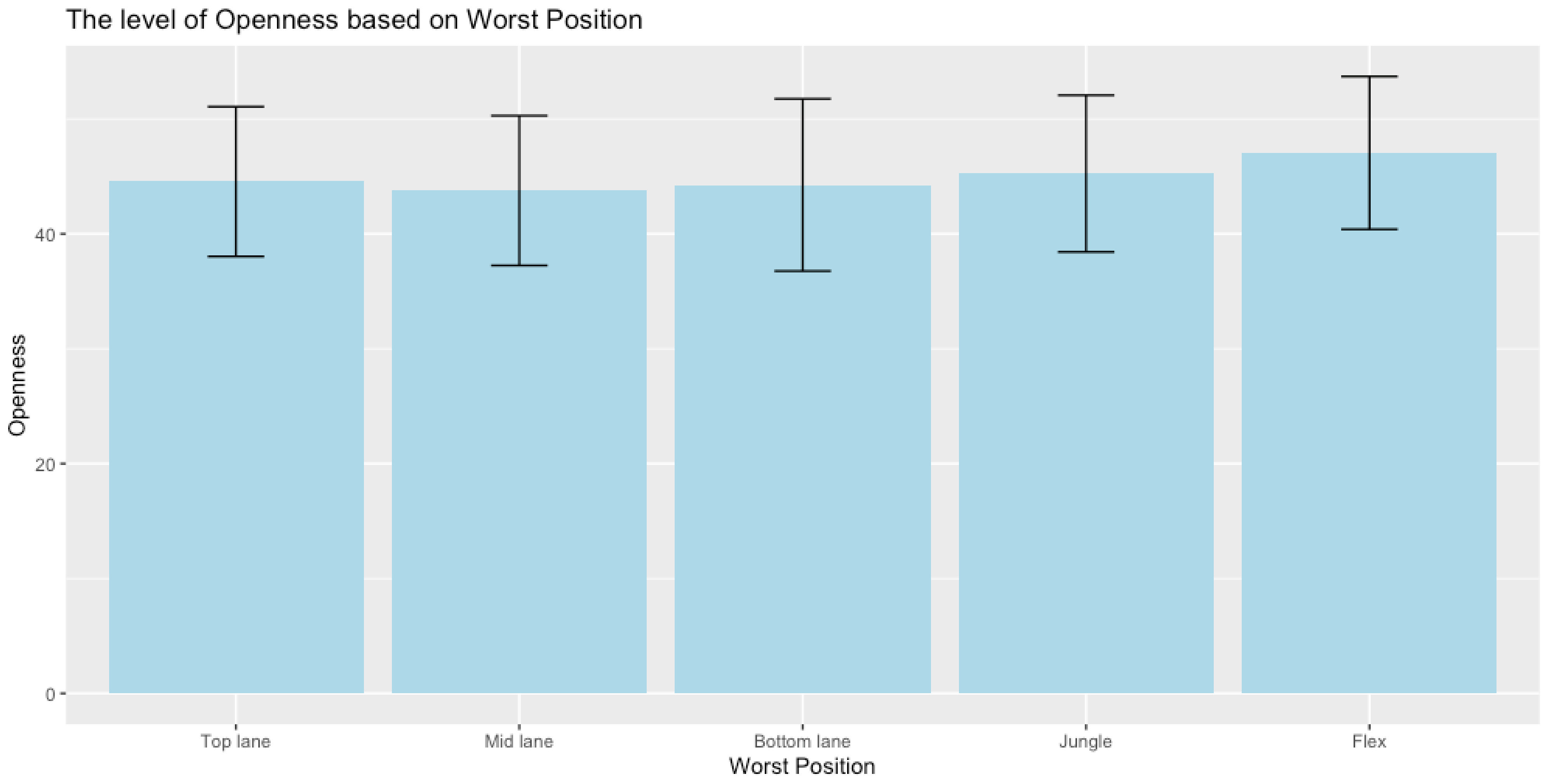
Data exploration



Data exploration

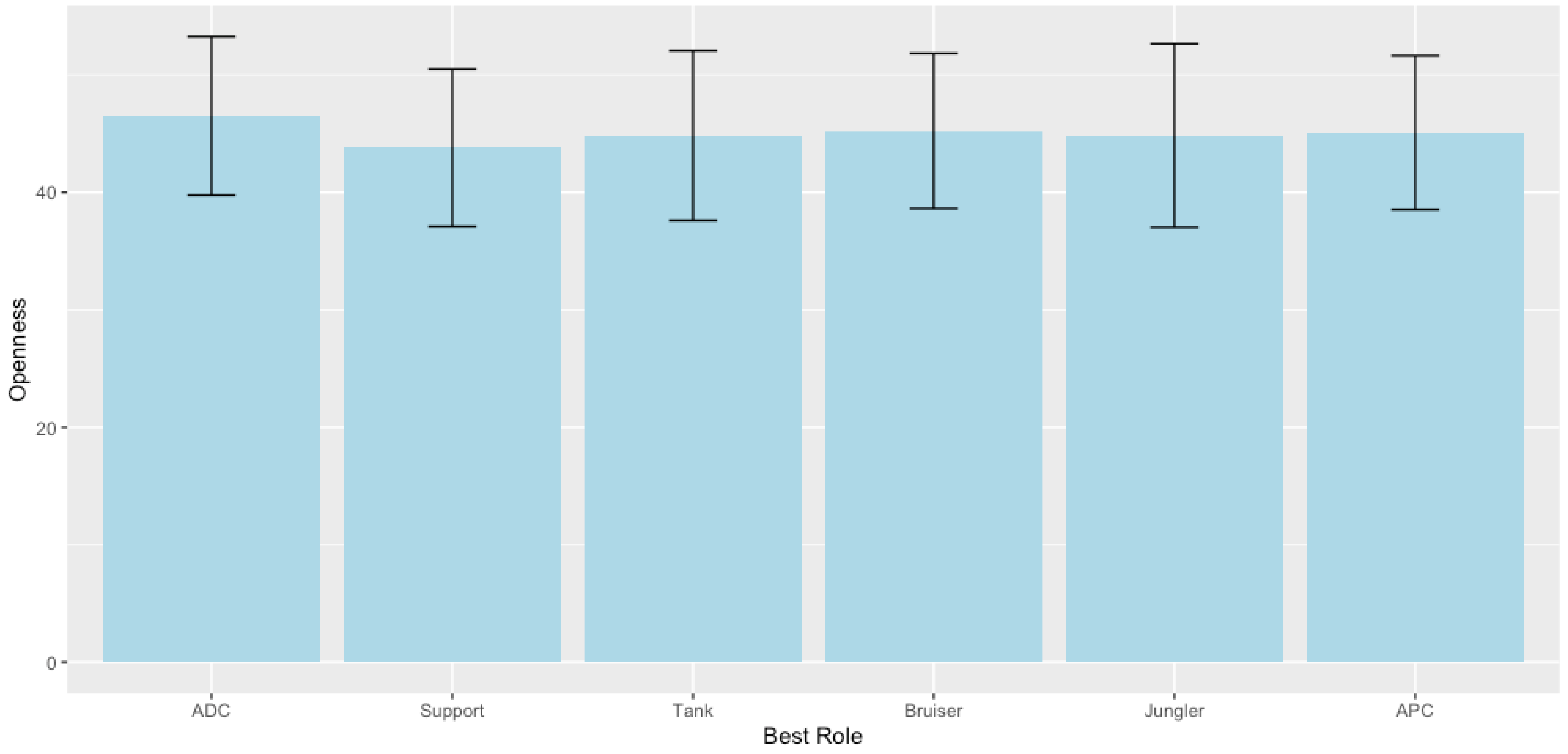


Data exploration

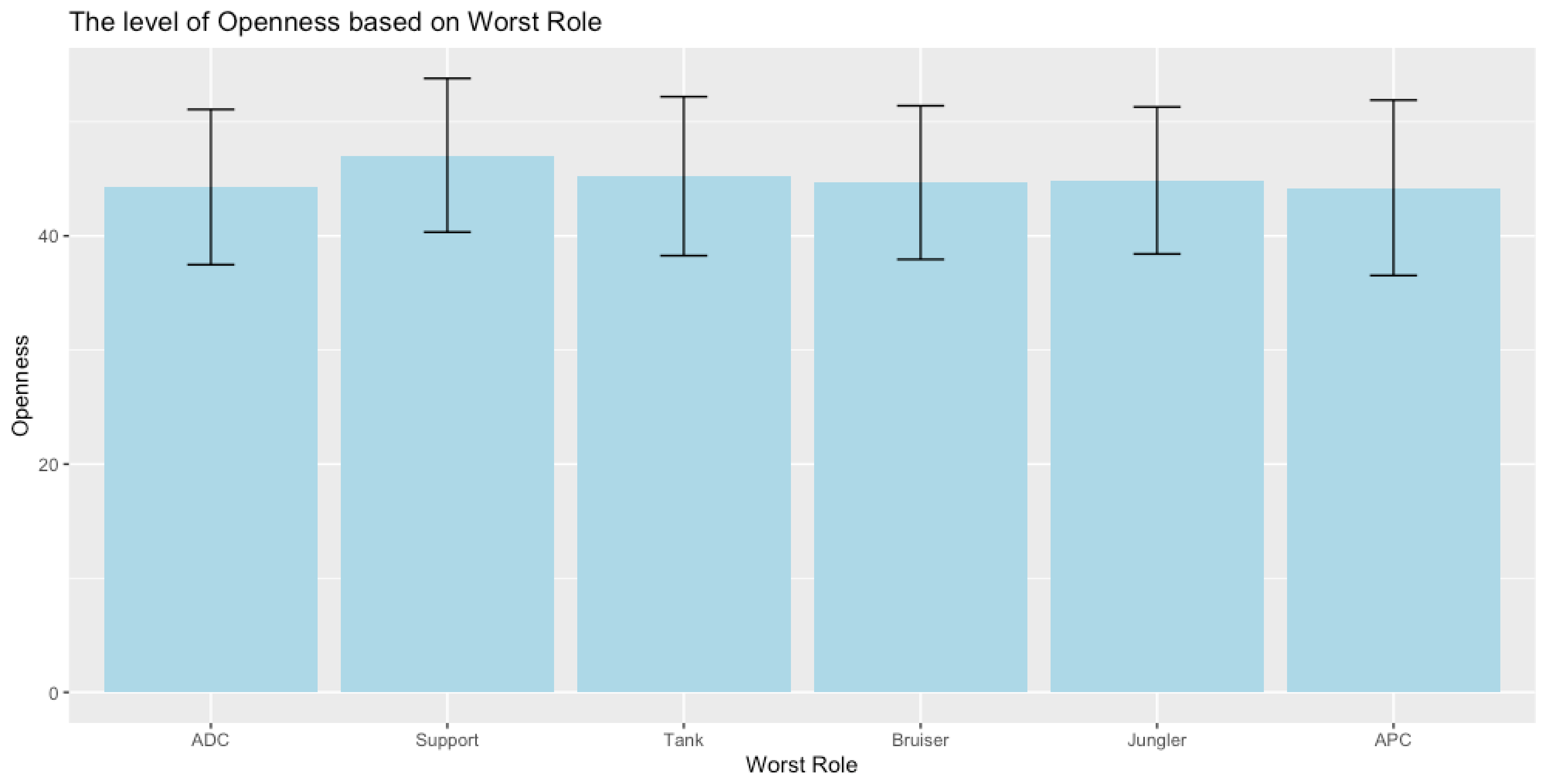


Data exploration

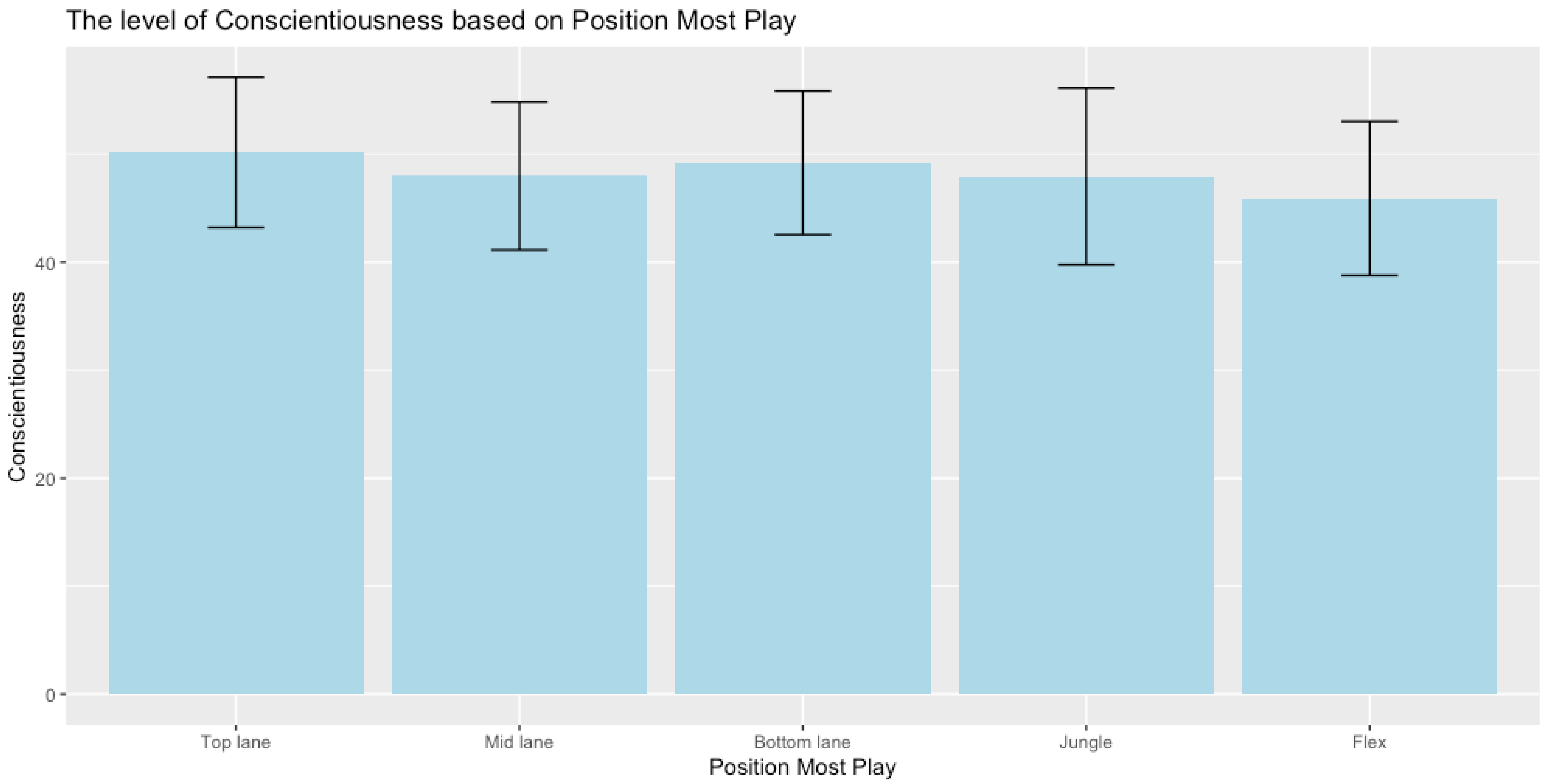
The level of Openness based on Best Role



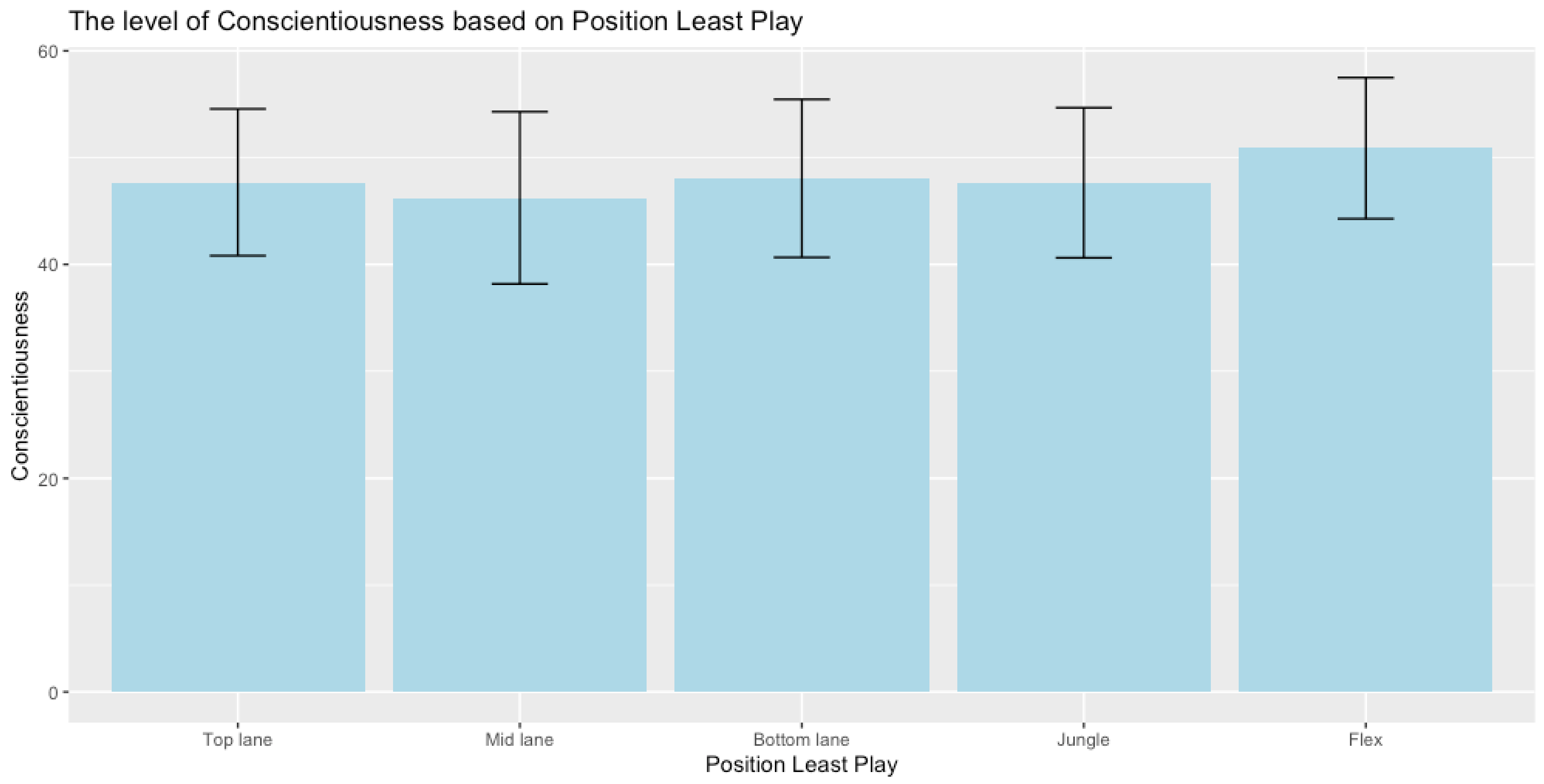
Data exploration



Data exploration



Data exploration



Data exploration

The class distribution of the targets/labels of the dataset

	Neuroticism	Extraversion	Openness	Agreeableness	Conscientiousness
High	50.19%	50.09%	54.54%	55.20%	50.47%
Low	49.81%	49.91%	45.46%	44.80%	49.53%

```
df['Neuroticism'] = df['Neuroticism'].apply(lambda x: "low" if x < 28 else "high" )
df['Extraversion'] = df['Extraversion'].apply(lambda x: "low" if x < 47 else "high" )
df['Openness'] = df['Openness'].apply(lambda x: "low" if x < 46 else "high" )
df['Agreeableness'] = df['Agreeableness'].apply(lambda x: "low" if x < 43 else "high" )
df['Conscientiousness'] = df['Conscientiousness'].apply(lambda x: "low" if x < 50 else "high" )
```

Methodology

Data splitting:

- 75% training set
- 25% test set

Model evaluation:

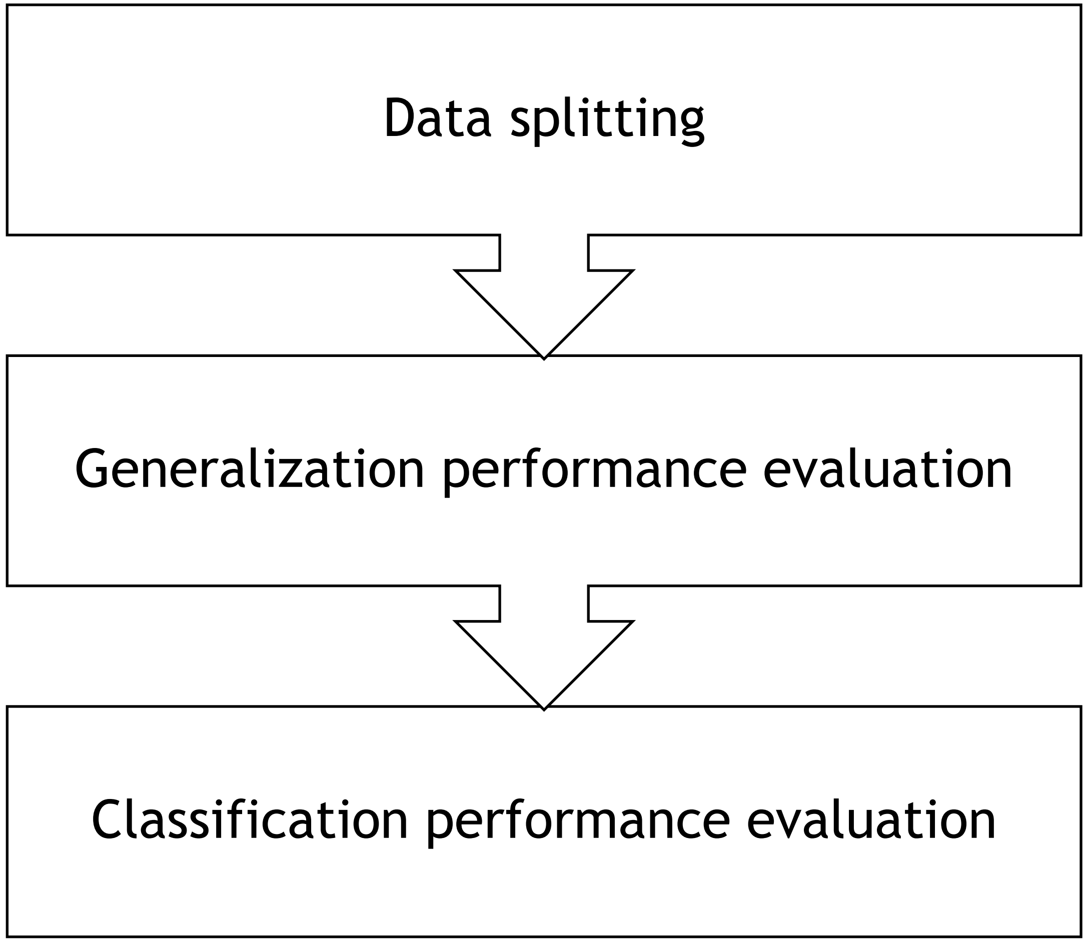
10-fold gridsearch cross-validation (CV)

10-fold CV (based on training set)

Hold-out method (based on test set)

Evaluation metric:

Macro F1 score


Data splitting
Generalization performance evaluation
Classification performance evaluation

Methodology—Evaluation Metric

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

$$Precison = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

$$F1\ score = 2\ *\ \frac{precision\ *\ recall}{precision + recall}$$

$$macro\ F1\ score = \frac{sum\ (F1\ scores)}{number\ of\ classes}$$

Methodology

The baseline model performance:

DummyClassifier(strategy="most_frequent"):

	Neuroticism	**Extraversion**	**Openness**	**Agreeableness**	**Conscientiousness**
Macro F1 score	0.33	0.33	0.35	0.36	0.34

Machine learning models used in the study:

A pipeline = feature encoder+ Classifier

Feature encoder: OneHotEncoder()

Classifiers:

- LogisticRegression()
- DecisionTreeClassifier()
- RandomForestClassifier()
- KNeighborsClassifier()
- SVC()

Model evaluation

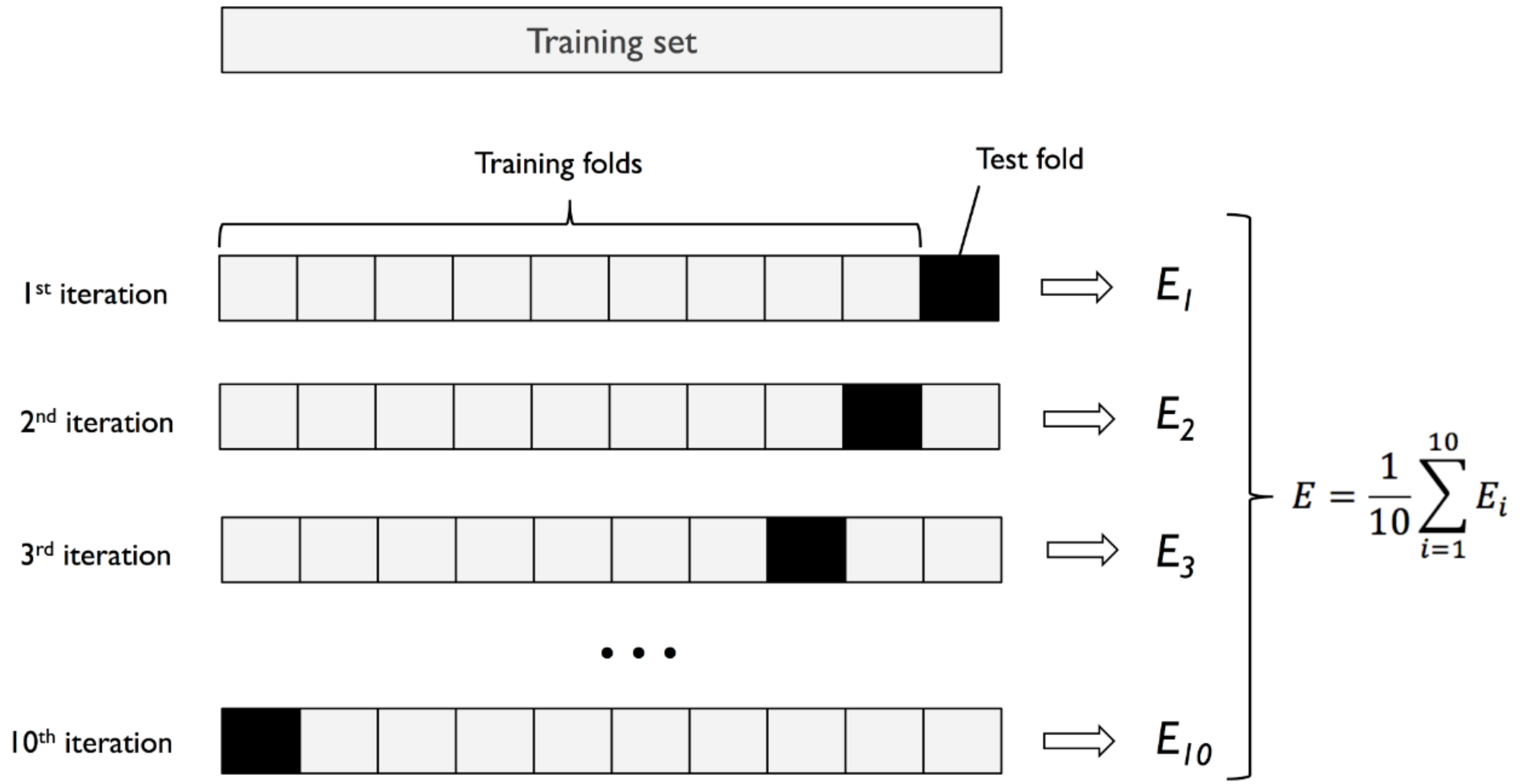
10-fold gridsearch cross-validation

Example:

```
lr = make_pipeline(OneHotEncoder(), LogisticRegression(class_weight='balanced', random_state=42))

lr_param_grid={'logisticregression__C': [0.01, 0.1, 1.0, 10, 100],
               'logisticregression__solver': ['newton-cg', 'sag', 'saga','lbfgs']}

lr_gs = GridSearchCV(lr, lr_param_grid, cv=10, scoring='f1_macro', n_jobs=-1, verbose=1, refit=True)
lr_gs.fit(X_train, y_train)

print(f"best score: {lr_gs.best_score_}")
print(f"best params: {lr_gs.best_params_}")
```

Model evaluation

10-fold cross-validation



Results

Generalization performance of models (10-fold CV, macro F1 score)

		Neuroticism	Extraversion	Openness	Agreeableness	Conscientiousness
LogisticRegression	CV mean	0.66	0.65	0.63	0.57	0.68
	CV sd	0.05	0.06	0.05	0.04	0.05
DecisionTree	CV mean	0.60	0.63	0.59	0.57	0.60
	CV sd	0.05	0.08	0.08	0.04	0.06
RandomForest	CV mean	0.68	0.64	0.65	0.56	0.68
	CV sd	0.06	0.06	0.07	0.06	0.05
K-Nearest Neighbor	CV mean	0.64	0.63	0.62	0.55	0.66
	CV sd	0.05	0.05	0.07	0.06	0.06
Support Vector Machine	CV mean	0.67	0.65	0.64	0.58	0.68
	CV sd	0.05	0.06	0.07	0.05	0.05

Results

Classification performance of models (the held-out test set, macro F1 score)

	Neuroticism	Extraversion	Openness	Agreeableness	Conscientiousness
LogisticRegression	0.66	0.64	0.66	0.57	**0.69**
DecisionTree	0.66	0.58	0.61	0.56	0.62
RandomForest	**0.75**	0.63	0.67	**0.60**	0.66
K-Nearest Neighbor	0.68	0.60	0.60	0.54	0.65
Support Vector Machine	0.72	**0.66**	**0.68**	0.58	0.68

Results

Which personality can be predicted?

All the 5 personalities can be predicted, especially Neuroticism achieving a macro F1 score of 0.75 as its highest classification performance by RandomForest;

Agreeableness achieves a macro F1 score of 0.60 as its highest, indicating that this personality is comparatively difficult to predict based on gaming behaviors, yet, still predictable.

Implications

This study could contribute to the literature on the complex relationship between gaming behavior and personality.

The results are meaningful for managerial practices such as staffing and managing.

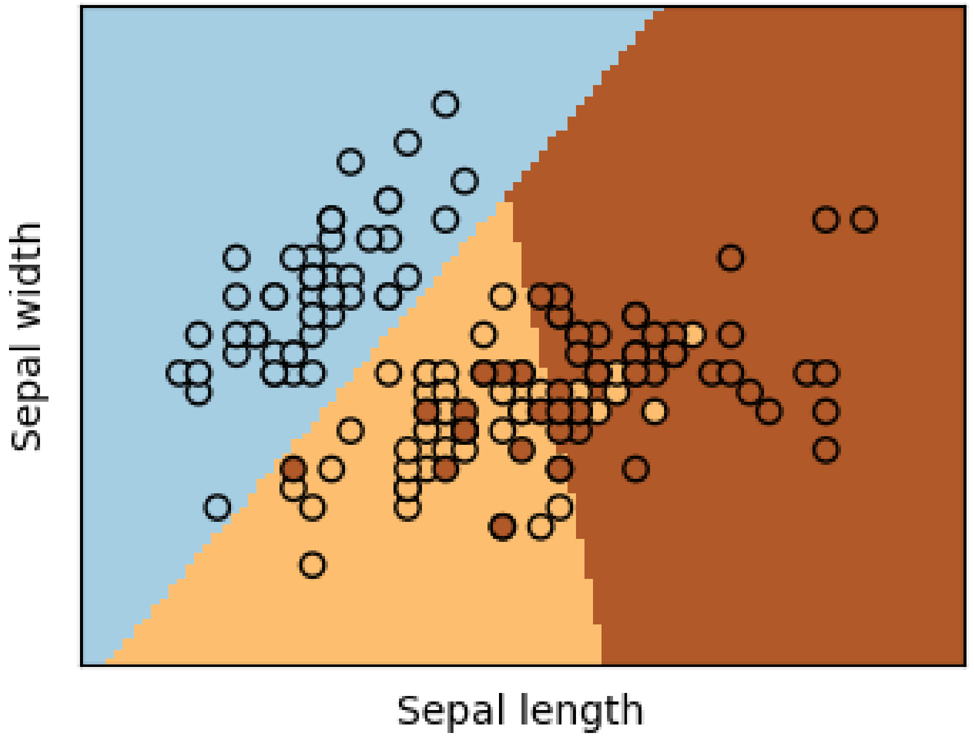
Next step

Which gaming behavior best predicts personality?

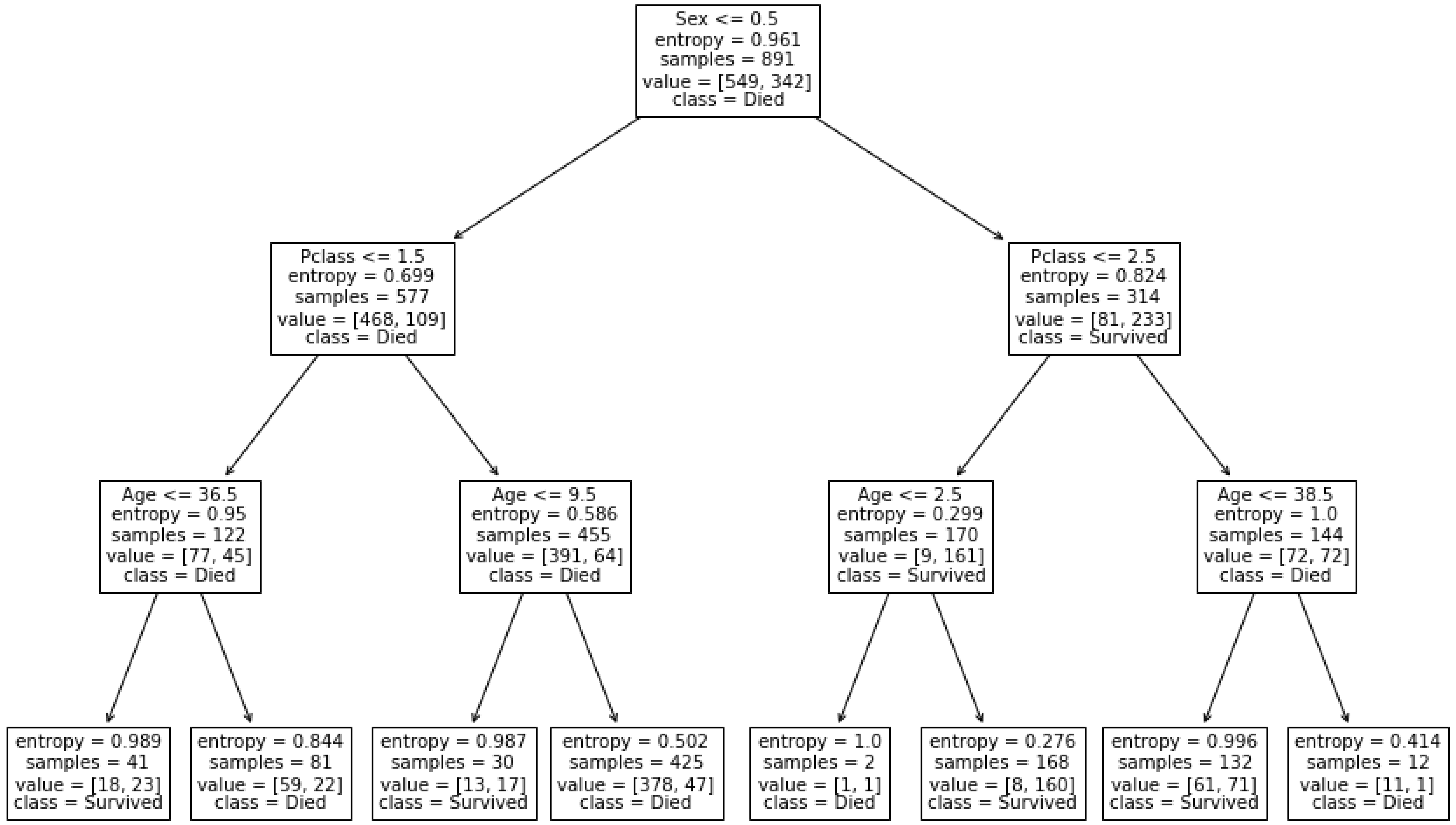
BI

AACSB ACCREDITED EQUIS ACCREDITED ASSOCIATION OF MBAS ACCREDITED

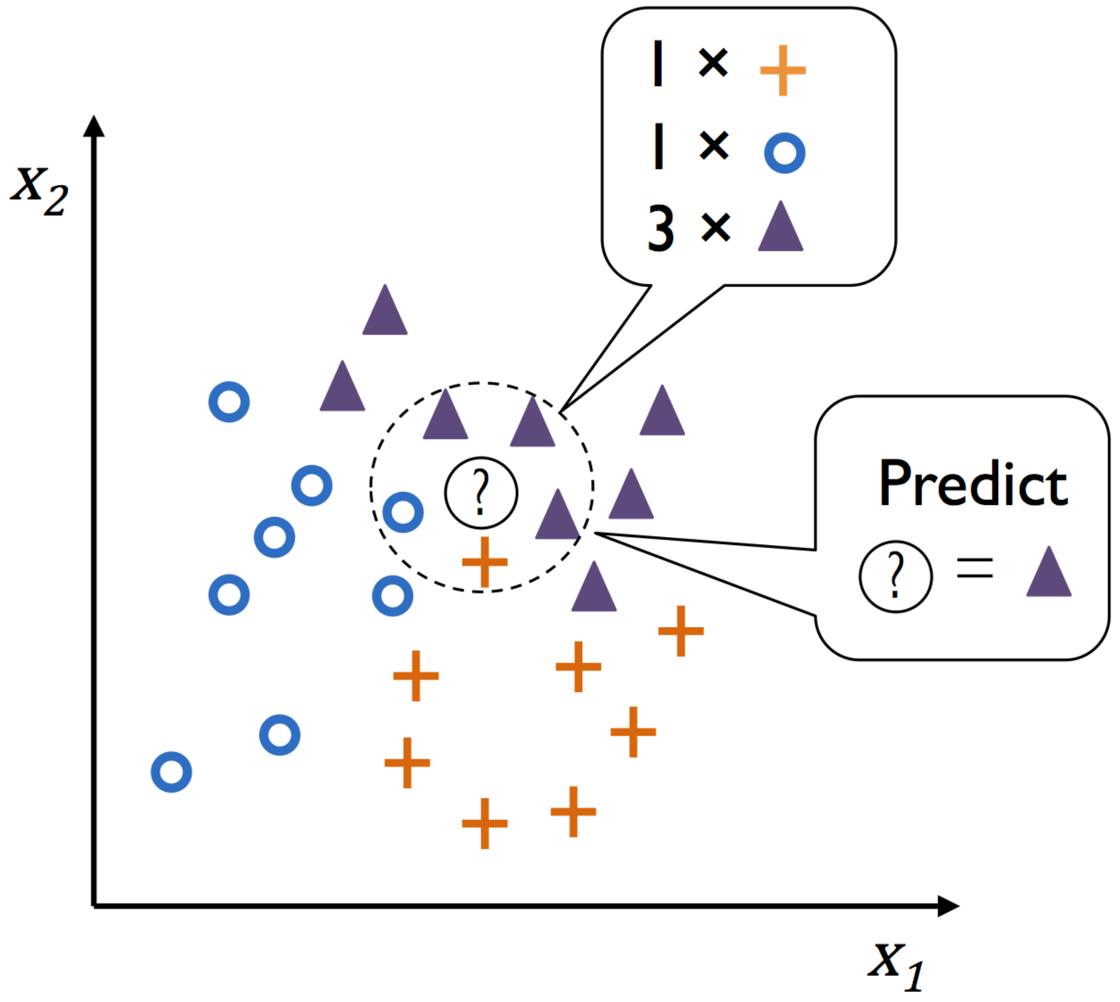
LogisticRegression



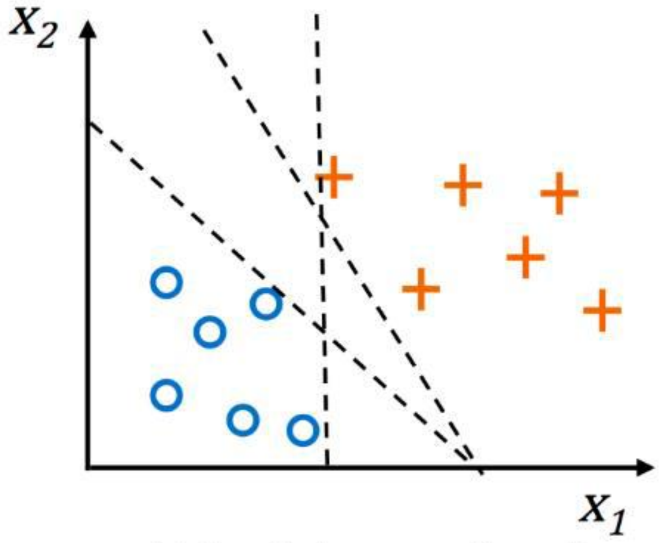
DecisionTreeClassifier



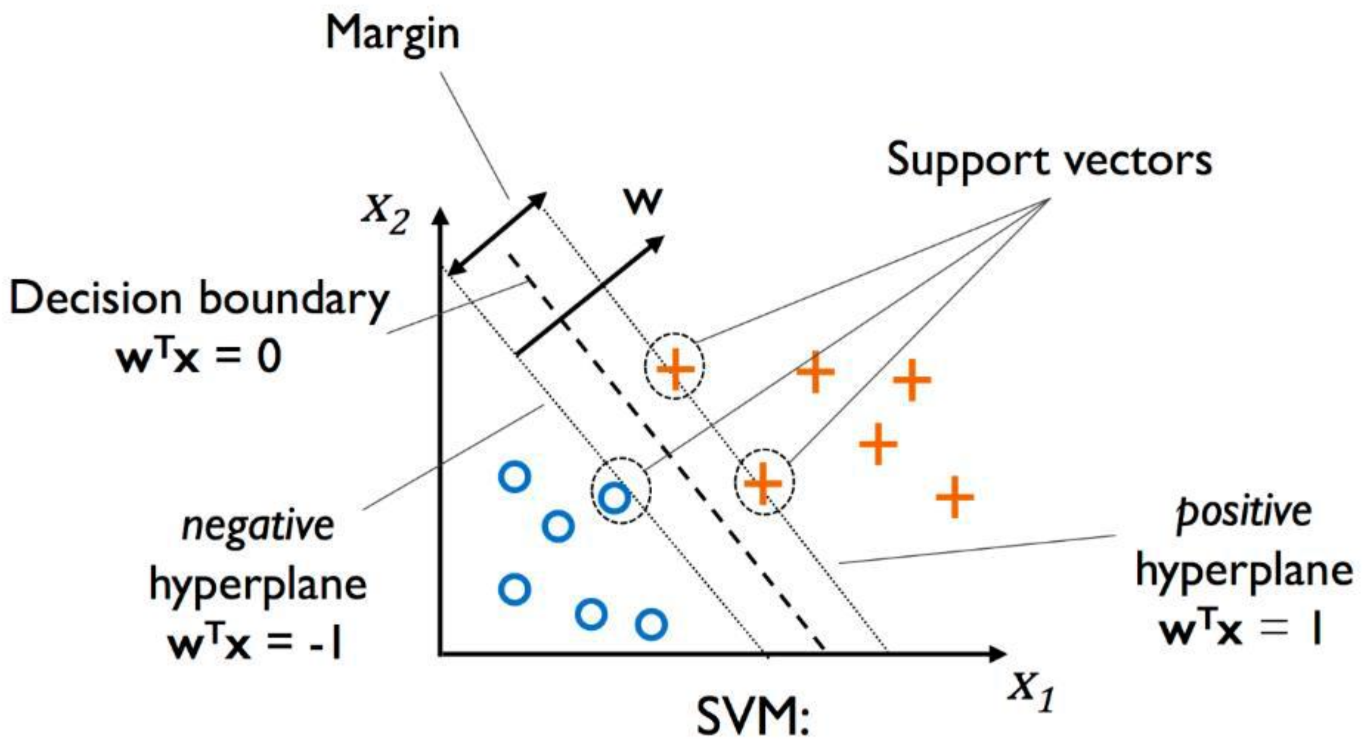
KNN



SVM



Which hyperplane?



SVM:
Maximize the margin