

# 1주차 2조

팀원: 황동욱, 권도혁

```
lookup.KeyValue  
f.constant(['em  
=tf.constant([0  
ce = tf.lookup.StaticV  
init,  
num_oov_buckets=5)
```

```
lookup.StaticVocabular  
initializer,  
num_oov_buckets,  
lookup_key_dtype=None  
name=None,  
experimental_is_open
```

# 자료 조사

# 소개

## Titanic - Machine Learning from Disaster: 추천 대회

### 배경:

- 연도는 2912년.
- "Spaceship Titanic"이라는 별간 여객선이 새롭게 살기에 적합한 세 개의 외계 행성으로 약 13,000명의 승객을 운송 중이었습니다.
- 선박은 알파 센타우리를 지나가는 도중에 시공간 이상 현상과 충돌하여 승객 중 거의 절반이 다른 차원으로 전송되었습니다.

### 목표:

- 참가자들은 손상된 우주선의 컴퓨터 시스템에서 복구된 기록을 사용하여 어떤 승객이 이상 현상에 의해 전송되었는지 예측해야 합니다.

### 세부 사항:

- 평가 기준: 분류 정확도.
- 제출 형식: 각 승객의 ID와 함께 그들이 전송되었는지 여부를 나타내는 부울 값이 포함된 CSV 파일.

예:

```
PassengerId,Transported
0013_01,False
0018_01,False
...
```

# Feature들의 의미 & 형태

## Feature 설명

이 대화의 목적은 "Spaceship Titanic"의 시공간 이상 현상과의 충돌 도중 승객이 다른 차원으로 전송되었는지 예측하는 것입니다. 이러한 예측을 돕기 위해, 손상된 선박의 컴퓨터 시스템에서 복구된 개인 기록이 제공됩니다.

파일 및 데이터 필드 설명:

1. **train.csv** - 승객들 중 약 2/3인 (~8700명)의 개인 기록. 이 데이터는 훈련 데이터로 사용됩니다.

Categorical

- **PassengerId(object)**: 각 승객에 대한 고유 ID. 'gggg\_pp' 형식을 가집니다. 여기서 'gggg'는 승객이 함께 여행하는 그룹을 나타내며, 'pp'는 그 그룹 내에서의 승객 번호입니다. 그룹의 사람들은 종종 가족이지만, 항상 그런 것은 아닙니다.
- **HomePlanet(object)**: 승객이 출발한 행성. 일반적으로 그들의 영구 거주지 행성입니다.
- **CryoSleep(object)**: 승객이 여정 동안 무의식 상태(동면)에 놓여 있었는지 여부. 크라이오슬립 상태의 승객은 객실에 제한됩니다.
- **Cabin(object)**: 승객이 머무르는 객실 번호. 'deck/num/side' 형식을 가집니다. 여기서 'side'는 Port(P) 또는 Starboard(S) 일 수 있습니다.
- **Destination(object)**: 승객이 하차할 행성.
- **VIP(object)**: 승객이 여정 동안 특별한 VIP 서비스를 위해 지불했는지 여부.
- **Name(object)**: 승객의 이름.

Numeric

- **Age(float64)**: 승객의 나이.
- **RoomService(float64), FoodCourt(float64), ShoppingMall(float64), Spa(float64), VRDeck(float64)**: "Spaceship Titanic"의 다양한 호화 시설에서 승객이 청구한 금액.

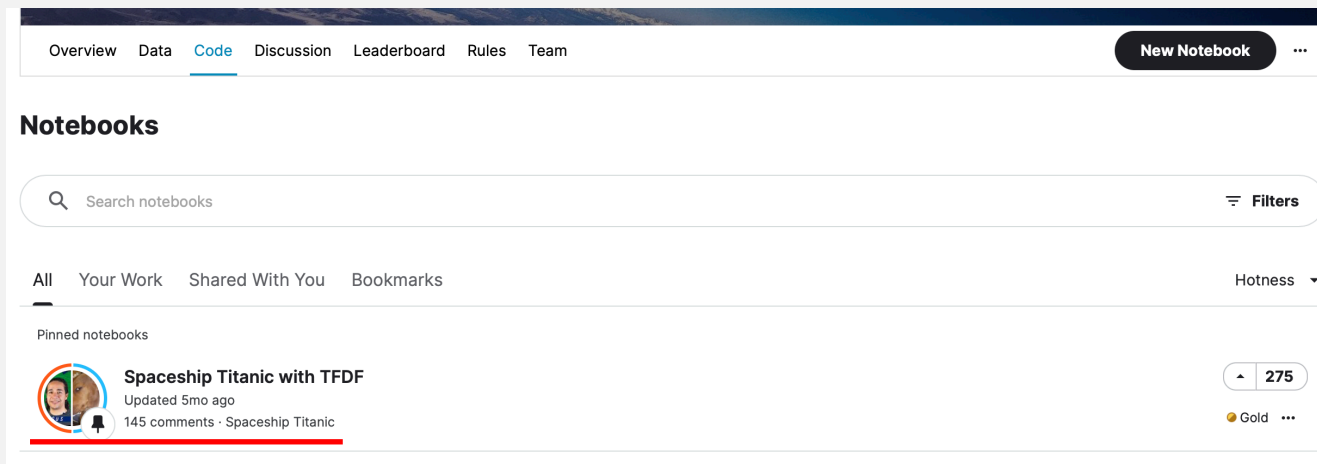
## Feautre들의 의미 & 형태

### Taget (Categorical)

- **Transported(bool)**: 승객이 다른 차원으로 전송되었는지 여부. 이것이 예측하려는 대상입니다.
2. **test.csv** - 승객 중 나머지 1/3인 (~4300명)의 개인 기록. 이 데이터는 테스트 데이터로 사용됩니다. 이 데이터 세트의 승객에 대해 '**Transported**' 값을 예측해야 합니다.
  3. **sample\_submission.csv** - 올바른 형식의 제출 파일.
    - **PassengerId**: 테스트 세트의 각 승객에 대한 ID.
    - **Transported**: 대상. 각 승객에 대해 True 또는 False를 예측합니다.

# 기술 스택 확인

## 모델 - kaggle code



출처: <https://www.kaggle.com/code/gusthema/spaceship-titanic-with-tfdf>

## 모델 - kaggle code

tensorflow

Accuracy: 0.8126

```
✓ import tensorflow as tf  
import tensorflow_decision_forests as tfdf
```

```
rf = tfdf.keras.RandomForestModel()  
rf.compile(metrics=["accuracy"])
```





## 모델 - kaggle code

[Overview](#) [Data](#) [Code](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [Submissions](#) [Submit Predictions](#) [...](#)

### Leaderboard

[Raw Data](#) [Refresh](#)

This leaderboard is calculated with all of the test data.

#	Team	Members	Score	Entries	Last	Join
1	CADang		0.95464	320	12h	
2	MLCV		0.92003	600	7h	

출처: <https://www.kaggle.com/code/gusthema/spaceship-titanic-with-tfdf>

## 모델 - kaggle code

Scikit—learn (sklearn)

Accuracy: 0.75

```
#Common Model Algorithms
from sklearn import svm, tree, linear_model, neighbors, naive_bayes, ensemble, discriminant_analysis, gaussian_process
from xgboost import XGBClassifier      "xgboost": Unknown word.
```

## 모델 - GPT 4.0

### 모델 추천

주어진 문제는 승객들이 다른 차원으로 이동했는지 아닌지를 예측하는 **이진 분류 문제**입니다.

#### 1. TensorFlow:

- **Deep Neural Network (DNN):** TensorFlow에 내장된 `tf.keras` API를 사용하여 깊은 신경망을 구축하고 훈련할 수 있습니다. DNN은 일반적으로 대규모 데이터셋에서 잘 작동합니다.
- **TensorFlow Decision Forests (TFDF):** 이는 이전에 언급한 방식입니다. 랜덤 포레스트나 그래디언트 부스팅 트리와 같은 트리 기반의 알고리즘을 사용하여 문제를 해결할 수 있습니다.

#### 2. Scikit-learn:

- **RandomForestClassifier:** 랜덤 포레스트는 여러 개의 의사결정 트리를 결합한 방식으로, 오버피팅을 방지하면서 다양한 데이터셋에서 잘 작동합니다.
- **GradientBoostingClassifier:** 이 방식은 강력한 트리 기반의 알고리즘이며, 순차적으로 트리를 구축하여 성능을 향상시킵니다.
- **Logistic Regression:** 데이터가 선형으로 분리될 수 있다고 생각되는 경우 사용될 수 있는 간단한 모델입니다.

## 모델 - GPT 4.0

### 3. PyTorch:

- **Custom Neural Networks:** PyTorch는 딥러닝 모델을 구축하고 훈련하기 위한 많은 유연성을 제공합니다. 특히 복잡한 데이터셋이나 문제에 대해 깊은 신경망을 사용할 때 유용합니다.
- **Pre-trained models with transfer learning:** PyTorch는 torchvision 라이브러리를 통해 미리 훈련된 모델을 제공합니다. 이를 기반으로 전송 학습을 사용하여 이 문제를 해결할 수 있습니다.

추천:












- 시작 단계에서는 **Scikit-learn**의 **RandomForestClassifier**로 시작하는 것을 추천합니다. 이는 구현이 간단하며 대부분의 데이터셋에서 좋은 성능을 제공합니다.
- 성능 향상이 필요한 경우 TensorFlow나 PyTorch를 사용하여 더 깊은 신경망을 구축해 보세요.

마지막으로, 어느 모델을 선택하든, 항상 데이터 전처리, 특성 엔지니어링, 모델 튜닝 등의 과정을 통해 성능을 최적화할 수 있습니다.

# 목표 설정

# 목표 설정

## Leaderboard

35	Lafoi		0.81529	3	13d
36	W3M		0.81529	2	5d
37	xasiban		0.81529	2	32m
38	✧ Tesla, Inc. ✧		0.81482	1	2mo
39	nlgm		0.81458	16	2mo
40	power overwhelming		0.81458	3	2mo
41	Aaditya Tyagi		0.81458	6	1mo
42	Kyle Cubit		0.81458	5	17d
43	luiscadi		0.81412	53	16d
44	Akshay Bhandari		0.81388	8	1mo
45	clayton21		0.81365	36	5d



목표 : 0.814

# 일정 계획

## 일정 계획

# Spaceship Titanic

전반부터 후반까지 계획

9월 8일

1차 회의

9월 10일

2차 회의  
모델 조사  
코드리뷰

9월 13일

3차 회의  
레포 만들기  
전처리

9월 15일

4차 회의  
모델 설계  
모델 훈련  
최적화

9월 17일

5차 회의  
최적화2

9월 20일

6차 회의  
최종 검토  
Kaggle 제출



# 협업 환경

The screenshot shows the GitHub interface for the repository 'gdsc\_kaggle / SpaceshipTitanic'. The left sidebar displays the file tree with the following structure:

- main
- SpaceshipTitanic
  - spaceship-titanic
  - 코드리뷰
    - best-models-spaceship-titanic.ipynb
    - spaceship-titanic-with-tfidf.ipynb
    - installation.ipynb
    - main.ipynb
    - spaceship-titanic.zip

The main content area shows the commit history for the 'SpaceshipTitanic' directory. The commit is by 'BrendanHwnag' with the message 'back up' and hash 'c0b93d4' from 3 days ago. The commit history table is as follows:

Name	Last commit message	Last commit date
..		
spaceship-titanic	back up	3 days ago
코드리뷰	back up	3 days ago
installation.ipynb	back up	3 days ago
main.ipynb	back up	3 days ago
spaceship-titanic.zip	back up	3 days ago

[Overview](#) [Data](#) [Code](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#)[Submissions](#)[Submit Predictions](#)

...

TEAM NAME

GDSC Hanyang ML/DL 2조

This name will appear on your team's leaderboard position.

## Let others know you're looking for teammates

[See tips about forming teams](#)To use this feature, you need to reach the Contributor level on Kaggle. [Learn more](#)Display your status with  on the leaderboard

## Team Members

Your team can have a maximum of 10 members.

**dongughwang (You)**

Team Leader

**□Dohyeok Kwon**

Member

[Send Invitation](#)

# Notion

Spaceship Titanic

공유 💬 ⌚ ☆ ...

## Spaceship Titanic

소개

Feature 설명

GPT가 해준 모델 추천

1. TensorFlow:
2. Scikit-learn:
3. PyTorch:

알고리즘

1. Random Forest와 Gradient Boosted Trees 의 주된 차이

기법 & 전략

1. 결측치 채우기
2. 상관계수
3. Tree based 모델 전처리

궁금한거

- 🤔 Random Forest는 왜 손실함수가 없지?
- 🤔 tfidf를 쓰게 되면 feature scaling은 필요 없나?
- 🤔 그럼, 다른 모델들은? feature scaling 해야 하는지...
- 🤔 그럼, scaling을 해서 나쁜건 없다?
- 🤔 skewness를 조정하는 건 어떤 모델에 대해서 필요하지?
- 🤔 skewness를 조정해서 나쁜건 없다?

참고한 코드점검해야 하는 것들

?

# 프로토타입 개발

## 모델

```
model_classes = [tfdk.keras.RandomForestModel, tfdk.keras.GradientBoostedTreesModel, tfdk.keras.CartModel]
```

## 결측치 확인 & Imputation

### 결측치 비율 상세 확인

RoomService와 Total\_Service는 처리 필요

```
missing_ratio = (dataset_df.isnull().sum() / len(dataset_df)) * 100  
missing_ratio_sorted = missing_ratio.sort_values(ascending=False)  
missing_ratio_sorted
```

✓ 0.0s

CryoSleep	2.496261
ShoppingMall	2.392730
VIP	2.335212
HomePlanet	2.312205
Cabin	2.289198
VRDeck	2.162660
FoodCourt	2.105142
Spa	2.105142
Destination	2.093639
RoomService	2.082135
Age	2.059128
Transported	0.000000
Group	0.000000

dtype: float64

```
] = dataset_df[['VIP', 'CryoSleep', 'FoodCourt', 'ShoppingMall', 'Spa', 'VRDeck']].fillna(value=0)
```

## Feature 수정

```
# PassengerId(object): 각 승객에 대한 고유 ID. 'gggg_pp' 형식을 가집니다. 여기서 'gggg'는 승객이 함께 여행하는 그룹을 나타내며,  
# 'pp'는 그 그룹 내에서의 승객 번호입니다. 그룹의 사람들은 종종 가족이지만, 항상 그런 것은 아닙니다.  
# PassengerId를 gggg로 나누어 그룹 ID를 얻어냅니다.
```

(parameter) x: Any

```
dataset_df['Group'] = dataset_df['PassengerId'].apply(lambda x: x.split('_')[0])
```

```
# int로 변환
```

```
dataset_df['Group'] = dataset_df['Group'].astype(int)      "astype": Unknown word.
```



## Feature 형변환

```
label = "Transported"
dataset_df[label] = dataset_df[label].astype(int)      "astype": U
dataset_df['VIP'] = dataset_df['VIP'].astype(int)      "astype": U
dataset_df['CryoSleep'] = dataset_df['CryoSleep'].astype(int)
```

✓ 0.0s

## Feature 생성(쪼개기)

- **Cabin(object):** 승객이 머무르는 객실 번호. 'deck/num/side' 형식을 가집니다. 여기서 'side'는 Port(P) 또는 Starboard(S) 일 수 있습니다.

```
dataset_df[["Deck", "Cabin_num", "Side"]] = dataset_df["Cabin"].str.split("/", expand=True)
```

## 결과

```
RandomForestModel: 0.7987  
GradientBoostedTreesModel: 0.7956  
CartModel: 0.7619
```

감사합니다