SPACESHIP TITANIC

*#scoatem "Cabin", "Name", "Transported"*

cols = ["PassengerId", "HomePlanet", "CryoSleep", "Destination", "Age", "VIP", "RoomService", "FoodCourt", "ShoppingMall", "Spa", "VRDeck"]

X\_train = train\_data[cols]

y\_train = train\_data.Transported

*#pt ca aici erau cu true si false, de obicei nu tb sa fac asta*

y\_train = y\_train.astype(int)

X\_valid = test\_data[cols]

In [4]:

from sklearn.preprocessing import OneHotEncoder

object\_cols = ["HomePlanet", "CryoSleep", "Destination", "VIP"]

*# Apply one-hot encoder to each column with categorical data*

OH\_encoder = OneHotEncoder(handle\_unknown='ignore', sparse=False)

OH\_cols\_train = pd.DataFrame(OH\_encoder.fit\_transform(X\_train[object\_cols]))

OH\_cols\_valid = pd.DataFrame(OH\_encoder.transform(X\_valid[object\_cols]))

*# One-hot encoding removed index; put it back*

OH\_cols\_train.index = X\_train.index

OH\_cols\_valid.index = X\_valid.index

*# Remove categorical columns (will replace with one-hot encoding)*

num\_X\_train = X\_train.drop(object\_cols, axis=1)

num\_X\_valid = X\_valid.drop(object\_cols, axis=1)

*# Add one-hot encoded columns to numerical features*

OH\_X\_train = pd.concat([num\_X\_train, OH\_cols\_train], axis=1)

OH\_X\_valid = pd.concat([num\_X\_valid, OH\_cols\_valid], axis=1)

*# Ensure all columns have string type*

OH\_X\_train.columns = OH\_X\_train.columns.astype(str)

OH\_X\_valid.columns = OH\_X\_valid.columns.astype(str)

In [5]:

from sklearn.impute import SimpleImputer

*# Imputation*

my\_imputer = SimpleImputer()

imputed\_X\_train = pd.DataFrame(my\_imputer.fit\_transform(OH\_X\_train))

imputed\_X\_valid = pd.DataFrame(my\_imputer.transform(OH\_X\_valid))

*# Imputation removed column names; put them back*

imputed\_X\_train.columns = OH\_X\_train.columns

imputed\_X\_valid.columns = OH\_X\_valid.columns

In [6]:

from xgboost import XGBRegressor

from sklearn.metrics import mean\_absolute\_error

model = XGBRegressor(n\_estimators=500, learning\_rate=0.05, random\_state=0)

model.fit(imputed\_X\_train, y\_train)

preds = model.predict(imputed\_X\_valid)

binary\_preds = (preds >= 0.5).astype(int)

true\_false\_preds = binary\_preds == 1

In [7]:

*#ai grija sa modifici aia cu survived transported etc*

output = pd.DataFrame({'PassengerId': test\_data.PassengerId, 'Transported': true\_false\_preds})

output.to\_csv('submission.csv', index=False)

print("Your submission was successfully saved!")

Your submission was successfully saved!

TITANIC

*''''*

*from xgboost import XGBRegressor*

*from sklearn.metrics import mean\_absolute\_error*

*# Function for comparing different approaches*

*def score\_dataset(X\_train, X\_valid, y\_train, y\_valid):*

*model = XGBRegressor(n\_estimators=500, learning\_rate=0.05, random\_state=0)*

*model.fit(X\_train, y\_train)*

*preds = model.predict(X\_valid)*

*return mean\_absolute\_error(y\_valid, preds)*

*'''*

Out[3]:

In [4]:

*#am scos direct aia cu numele si la ticket si la cabin - eventual sa dam drop si la fare?*

cols = ["PassengerId", "Pclass", "Sex", "Age", "SibSp", "Parch", "Fare", "Embarked"]

X\_train = train\_data[cols]

y\_train = train\_data.Survived

X\_valid = test\_data[cols]

In [5]:

*#OH\_encoder pe "Sex" si pe "Embarked"*

from sklearn.preprocessing import OneHotEncoder

object\_cols = ["Sex", "Embarked"]

*# Apply one-hot encoder to each column with categorical data*

OH\_encoder = OneHotEncoder(handle\_unknown='ignore', sparse=False)

OH\_cols\_train = pd.DataFrame(OH\_encoder.fit\_transform(X\_train[object\_cols]))

OH\_cols\_valid = pd.DataFrame(OH\_encoder.transform(X\_valid[object\_cols]))

*# One-hot encoding removed index; put it back*

OH\_cols\_train.index = X\_train.index

OH\_cols\_valid.index = X\_valid.index

*# Remove categorical columns (will replace with one-hot encoding)*

num\_X\_train = X\_train.drop(object\_cols, axis=1)

num\_X\_valid = X\_valid.drop(object\_cols, axis=1)

*# Add one-hot encoded columns to numerical features*

OH\_X\_train = pd.concat([num\_X\_train, OH\_cols\_train], axis=1)

OH\_X\_valid = pd.concat([num\_X\_valid, OH\_cols\_valid], axis=1)

*# Ensure all columns have string type*

OH\_X\_train.columns = OH\_X\_train.columns.astype(str)

OH\_X\_valid.columns = OH\_X\_valid.columns.astype(str)

/opt/conda/lib/python3.10/site-packages/sklearn/preprocessing/\_encoders.py:868: FutureWarning: `sparse` was renamed to `sparse\_output` in version 1.2 and will be removed in 1.4. `sparse\_output` is ignored unless you leave `sparse` to its default value.

warnings.warn(

In [6]:

*#print(OH\_X\_train)*

In [7]:

*#facem SimpleImputer pe "Cabin"*

from sklearn.impute import SimpleImputer

*# Imputation*

my\_imputer = SimpleImputer()

imputed\_X\_train = pd.DataFrame(my\_imputer.fit\_transform(OH\_X\_train))

imputed\_X\_valid = pd.DataFrame(my\_imputer.transform(OH\_X\_valid))

*# Imputation removed column names; put them back*

imputed\_X\_train.columns = OH\_X\_train.columns

imputed\_X\_valid.columns = OH\_X\_valid.columns

*#print(score\_dataset(imputed\_X\_train, imputed\_X\_valid, y\_train, y\_valid))*

In [8]:

from xgboost import XGBRegressor

from sklearn.metrics import mean\_absolute\_error

model = XGBRegressor(n\_estimators=500, learning\_rate=0.05, random\_state=0)

model.fit(imputed\_X\_train, y\_train)

preds = model.predict(imputed\_X\_valid)

binary\_preds = (preds >= 0.5).astype(int)

In [9]:

output = pd.DataFrame({'PassengerId': test\_data.PassengerId, 'Survived': binary\_preds})

output.to\_csv('submission.csv', index=False)

print("Your submission was successfully saved!")

Your submission was successfully saved!