titanic

March 18, 2021

[53]:
```
import pandas as pd
import numpy as np
import scipy
import lightgbm as lgb

from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_validate
from sklearn.linear_model import LogisticRegression
from sklearn import metrics

import matplotlib.pyplot as plt
import seaborn as sns
import optuna

pd.options.mode.chained_assignment = None
```

0.1 Ładowanie danych

Załadowanie danych dotyczących przeżywalności pasażerów najpopularniejszej katastrofy statku pasażerskiego na świecie. Dane ściągamy wykorzystując api udostępnione przez serwis kaggle.com .

[2]:
```
!kaggle competitions download -c titanic
```

```
titanic.zip: Skipping, found more recently modified local copy (use --force to
force download)
```

[3]:
```
!unzip -o titanic.zip
```

```
Archive:  titanic.zip
  inflating: gender_submission.csv
  inflating: test.csv
  inflating: train.csv
```

[52]:
```
# Dziele zbiór treningowy na trainingowy i testowy (zbiór testowy z kaggla nie␣
 ↪posiada zmiennej opisywanej)
df_train_all = pd.read_csv('./train.csv')
```

```
df_train, df_test = train_test_split(df_train_all, test_size=0.2,␣
↪random_state=42)
```

[5]:
```
df_train.head()
```

[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
331	332	0	1	Partner, Mr. Austen	male	45.5	0	0	113043	28.5000	C124	S
733	734	0	2	Berriman, Mr. William John	male	23.0	0	0	28425	13.0000	NaN	S
382	383	0	3	Tikkanen, Mr. Juho	male	32.0	0	0	STON/O 2. 3101293	7.9250	NaN	S
704	705	0	3	Hansen, Mr. Henrik Juul	male	26.0	1	0	350025	7.8542	NaN	S
813	814	0	3	Andersson, Miss. Ebba Iris Alfrida	female	6.0	4	2	347082	31.2750	NaN	S

0.2 Opis załadowanych featerów

Name	Description	Info
PassengerId	id	
Survived	Survival	0 = No, 1 = Yes
Pclass	Ticket	class 1 = 1st, 2 = 2nd, 3 = 3rd
Sex	Sex	
Age	Age in years	
SibSp	number of siblings / spouses aboard the Titanic	
Parch	number of parents / children aboard the Titanic	
Fare	Passenger fare	
Ticket	Ticket number	
Cabin	Cabin number	
Embarked	Port of Embarkation	C = Cherbourg, Q = Queenstown, S = Southampton

0.3 Cel

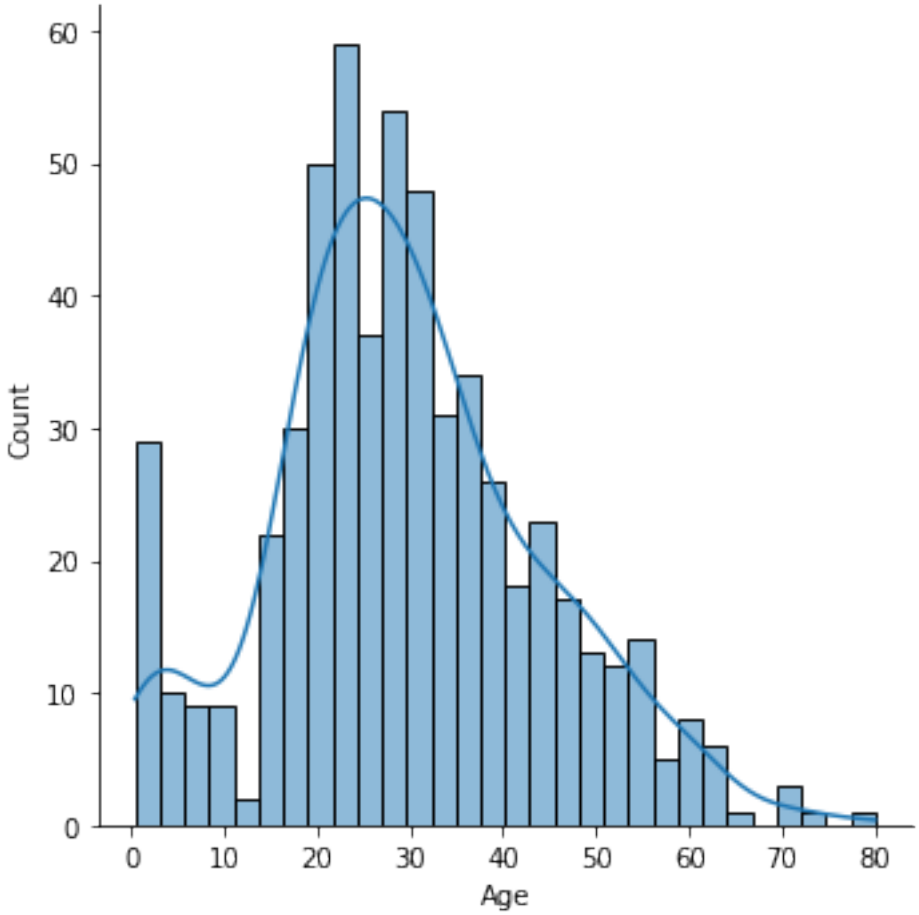
Naszym celem jest zbudowanie modelu, który przewidzi czy dana osoba przeżyje katastrofę. Wykorzystamy w tym celu bibliotekę LighGBM wykorzystującą technike gradient boosting tree.

0.4 Analiza danych

Celem sekcji jest powierzchowne zapoznanie się z danymi.

0.4.1 Age

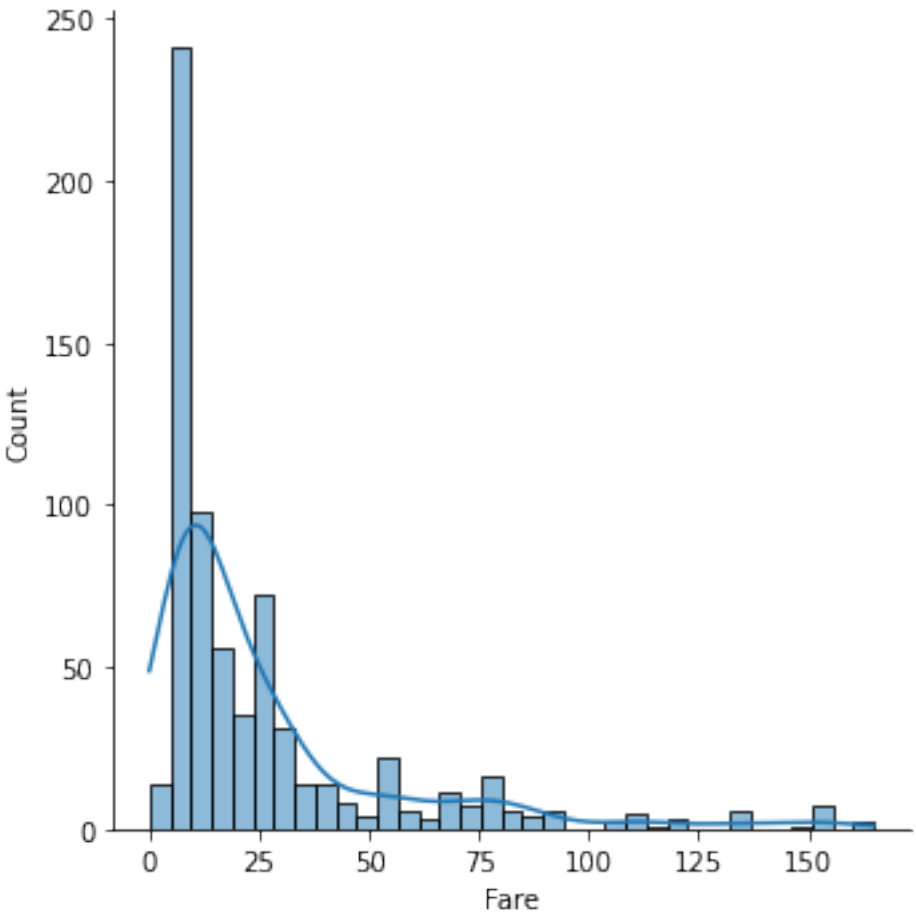
[18]:
```
sns.displot(df_train, x='Age', kde=True, kind='hist', bins=30)
plt.show()
```



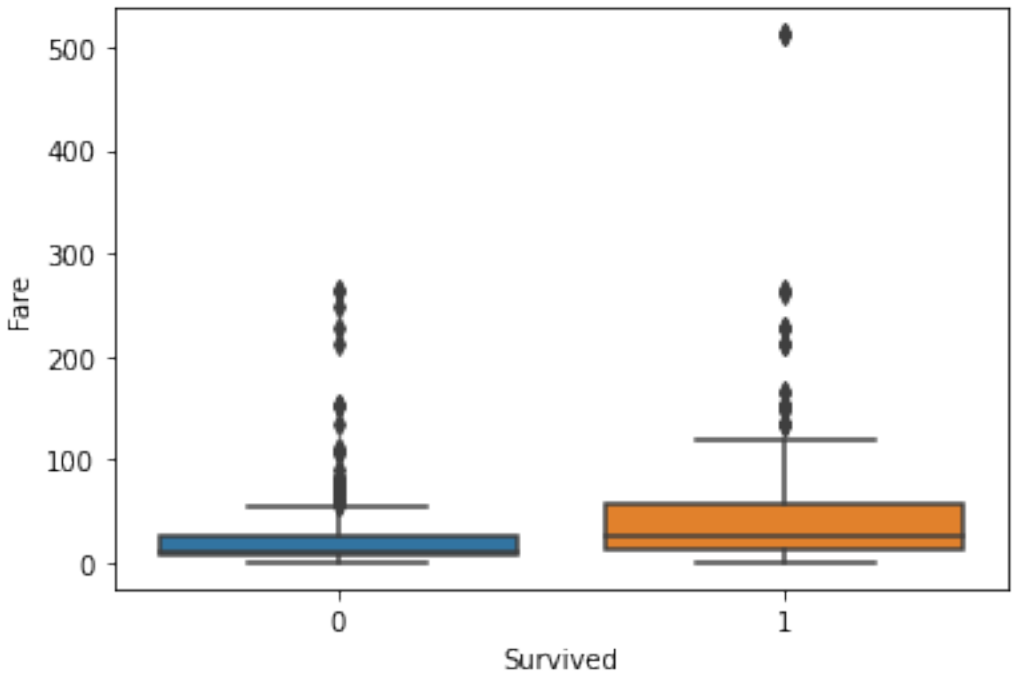
Patrząc na histogram, możemy zauważyć, że stosunkowo dużo jest najmłodszych dzieci/niemowlaków w porównaniu rozkładu wiekowego amerykańskiego społeczeństwa w tym okresie.

0.5 Fare

[34]:
```
sns.displot(df_train[df_train.Fare < 200], x='Fare', kde=True, kind='hist')
plt.show()
```



[20]:
```
sns.boxplot(y='Fare', x='Survived', data=df_train)
plt.show()
```



[36]:
```
scipy.stats.pointbiserialr(df_train['Survived'], df_train['Fare'])
```

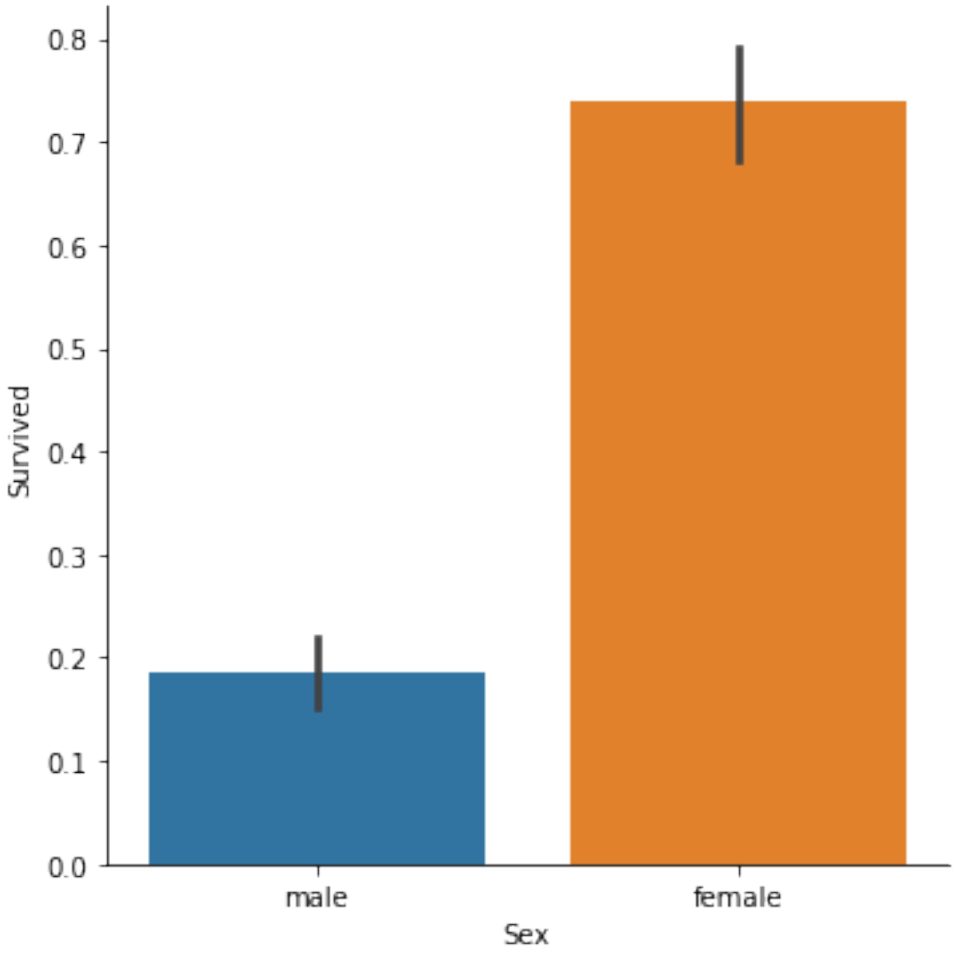
[36]:
```
PointbiserialrResult(correlation=0.2466407996192155,
pvalue=2.5064335809909983e-11)
```

Cena koreluje ze współczynnikiem 0.24 z opłatą za bilet.

0.5.1 Sex

[43]:
```
sns.factorplot(data=df_train, x="Sex", y="Survived", kind="bar")
plt.show()
```

```
/home/azapala/.local/lib/python3.8/site-packages/seaborn/categorical.py:3714:
UserWarning: The `factorplot` function has been renamed to `catplot`. The
original name will be removed in a future release. Please update your code. Note
that the default `kind` in `factorplot` (`'point'`) has changed `'strip'` in
`catplot`.
  warnings.warn(msg)
```



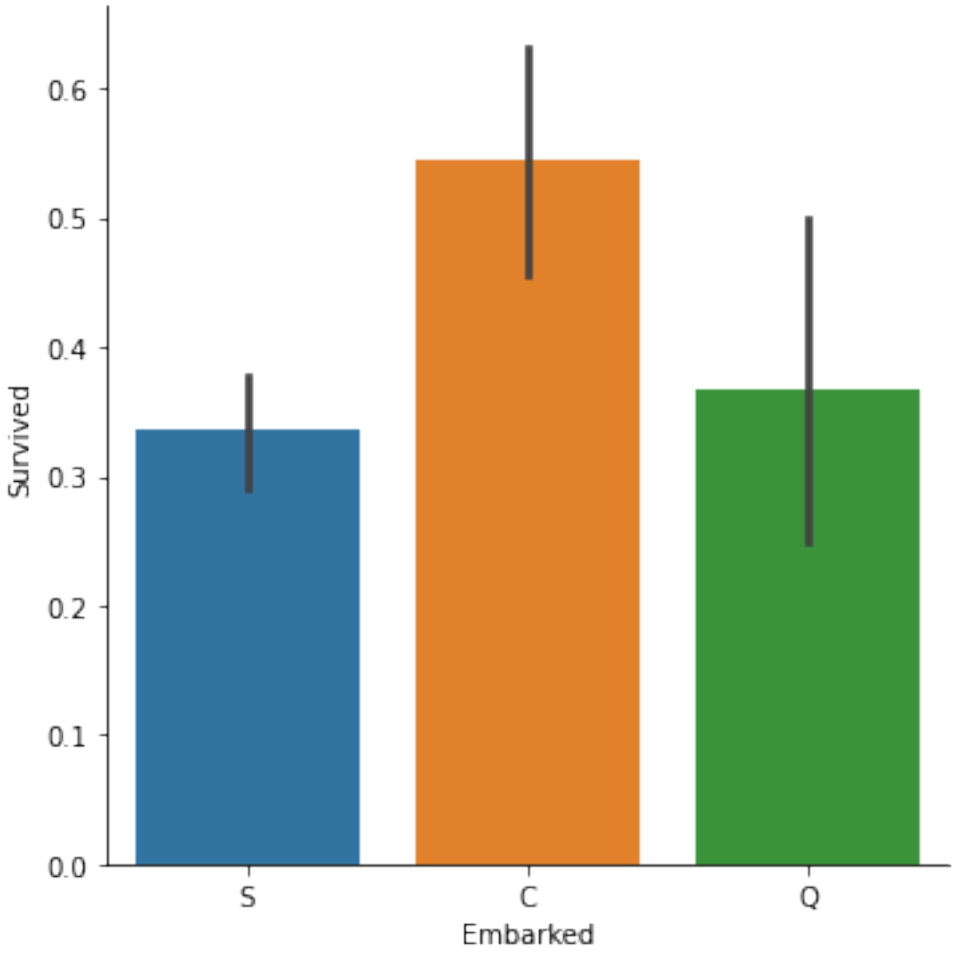
[9]:
```
df_train.groupby('Sex').size()
```

[9]:
```
Sex
female    245
male      467
dtype: int64
```

Kobiety w tym datasecie charakteryzują się znacznie więszką przeżywalnościa.

0.6 Embarked

[60]:
```
sns.catplot(data=df_train, x="Embarked", y="Survived", kind="bar")
plt.show()
```



[11]:
```
df_train.groupby('Embarked').size()
```

[11]:
```
Embarked
C    125
Q     60
S    525
dtype: int64
```

Wykorzystując tylko tą cechę stworzę model regressji logistycznej.

[58]:
```
embarked_model = LogisticRegression()
embarked_model.fit(pd.get_dummies(df_train['Embarked']), df_train['Survived'])
embarked_prediction = embarked_model.predict(pd.
 ↪get_dummies(df_test['Embarked']))
fpr, tpr, _ = metrics.roc_curve(df_test['Survived'], embarked_prediction)
metrics.auc(fpr, tpr)
```

[58]:
```
0.5832046332046331
```

Metryka auc modelu na zbiorze testowym jest > 0.5. Stąd ta tłumaczy cześć wariancji cechy przeżywalności, więc warto dodać port zaokrętowania do modelu.

0.6.1 Cechy ciągłe

Sprawdzę czy wiek, opłate za przejazd, SibSp i Parch można traktować jako zmienne ciągłe, dla małej liczby wartości można uwzględnić je jako cechy kategoryczne.

[12]:
```
df_train.Age.nunique()
```

[12]:
```
83
```

[13]:
```
df_train.Fare.nunique()
```

[13]:
```
220
```

[ ]:
```
Sprawdzę czy cechy
```

[14]:
```
df_train.groupby(['SibSp', 'Parch']).size().to_frame(name='counts')
```

[14]:

SibSp	Parch	counts
0	0	429
	1	31
	2	21
	3	1
	4	1
	5	1
1	0	96
	1	44
	2	18
	3	1
	4	2
	5	2
	6	1
2	0	14
	1	5
	2	3
	3	1
3	0	2
	1	6
	2	5
4	1	8
	2	8
5	2	5
8	2	7

Powyższe cechy mają sporo wartości stąd ciężko traktować je jako zmienne kategoryczne.

0.7 Dodanie dodatkowych cech

W tym paragrafie wykonam prostą inżynierie danych dodając proste cechy: * Liczba członków rodziny na pokładzie * Cecha binarna czy pasażer nie miał rodziny na pokładzie * Grupę wiekową

[15]:
```
for df in [df_train, df_test, df_train_all]:
    df['FamilySize'] = 0
    df.loc[:, 'FamilySize'] =  df['Parch'] + df['SibSp']
    df.loc['Single'] = 0
    df.loc[df['FamilySize'] > 0, 'Single'] = 1
```

[16]:
```
for df in [df_train, df_test, df_train_all]:
    df['AgeGroup'] = "baby"
    df.loc[(df.Age > 5) & (df.Age <= 14),'AgeGroup'] = "teenager"
    df.loc[(df.Age > 14) & (df.Age <= 60), 'AgeGroup' ] = "adult"
    df.loc[(df.Age > 60), 'AgeGroup' ] = "old"
```

0.8 Wybór kolumn do modelu

[17]:
```
model_columns = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked',␣
 ↪'FamilySize', 'Single', 'AgeGroup']

countinous_columns = ['Age', 'Parch', 'SibSp', 'FamilySize']
cat_columns = [col for col in model_columns if col not in countinous_columns]
```

[18]:
```
for dataset in [df_test, df_train, df_train_all]:
    dataset[cat_columns] = dataset[cat_columns].astype('category')
```

[19]:
```
df_train[model_columns].dtypes
```

[19]:
```
Pclass        category
Sex           category
Age            float64
SibSp            int64
Parch            int64
Fare          category
Embarked      category
FamilySize       int64
Single        category
AgeGroup      category
dtype: object
```

0.9 Trening

[20]:
```
def generate_lgb_dataset(df, model_columns, target_columns_name):
    return lgb.Dataset(df[model_columns], label=df[target_columns_name])
```

[21]:
```
d_train = generate_lgb_dataset(df_train, model_columns, 'Survived')
d_test =  generate_lgb_dataset(df_test, model_columns, 'Survived')

params={
    'application': 'binary',
    'metric': 'auc',
    'boosting': 'gbdt',
    'verbose': -1
}
```

0.9.1 CV

Sprawdzę metrykę auc dla 5-krotnej validacji krzyżowej.

[22]:
```
history = lgb.cv(params, d_train, num_boost_round=30, nfold=5)
```

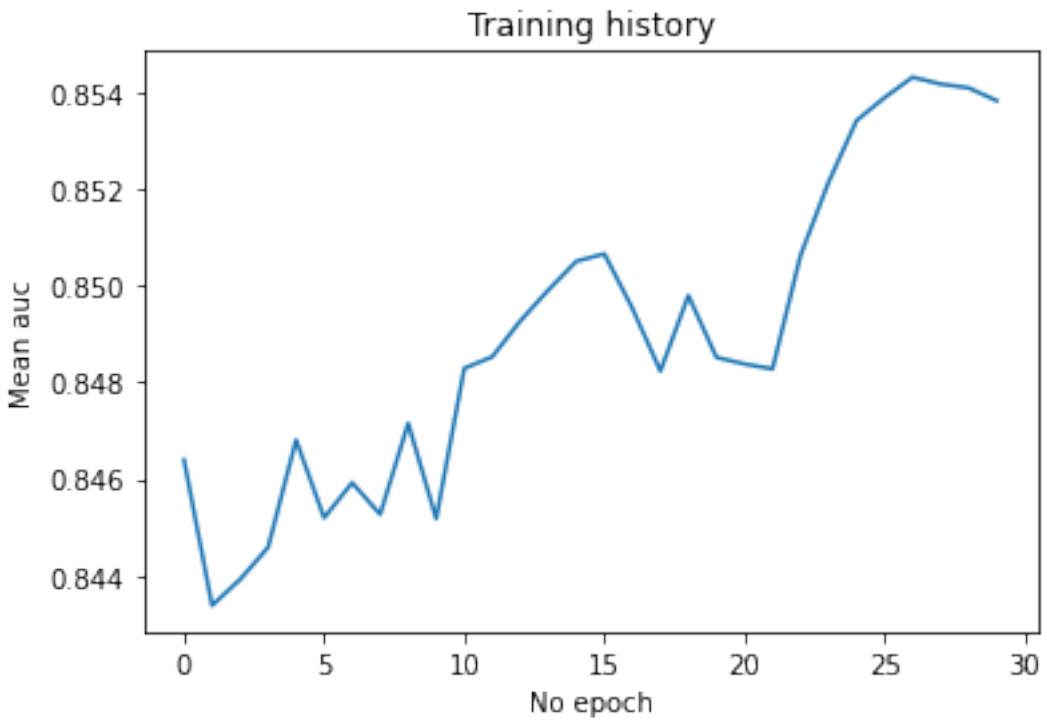
Mean auc metric based on 5-fold cross validation:

[23]:
```
history['auc-mean'][-1]
```

[23]:
```
0.853826192102639
```

[24]:
```
plt.plot(history['auc-mean'])
plt.title('Training history')
plt.xlabel('No epoch')
plt.ylabel('Mean auc')
```

[24]:
```
Text(0, 0.5, 'Mean auc')
```



Model osiąga **0.8538** średniego auc na 5-krotnej walidacji krzyżowej. Wykres wzrostu metryki nie wskazuje na przeuczenie.

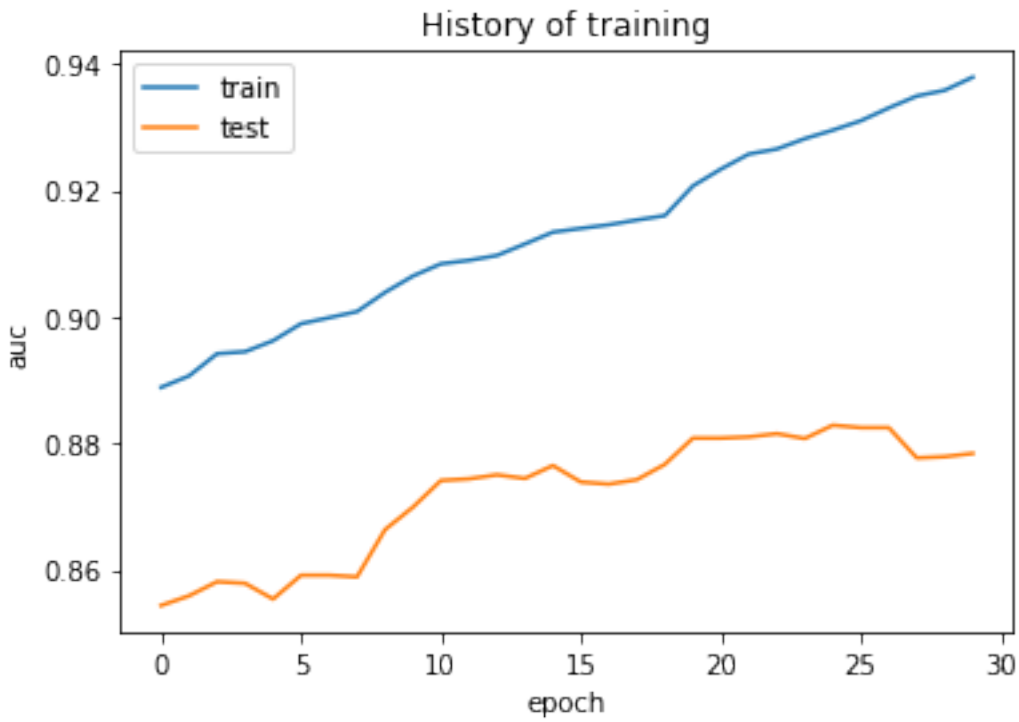
0.9.2 Test set

W tym paragrafie sprawdzę metrykę auc na wydzielonym datasecie testowym.

[25]:
```
def train_and_evaluate(params, d_train, d_test):
    history = {}
    bst = lgb.train(params, d_train, valid_sets=[d_train, d_test],␣
↪valid_names=['train', 'test'], evals_result=history, num_boost_round=30)

    plt.plot(history['train']['auc'])
    plt.plot(history['test']['auc'])
    plt.title('History of training')
    plt.xlabel('epoch')
    plt.ylabel('auc')
    plt.legend(['train','test'])
    plt.show()

    print(f"AUC: {history['test']['auc'][-1]}")
```

[26]:
```
train_and_evaluate(params, d_train, d_test)
```

```
/home/azapala/.local/lib/python3.8/site-packages/lightgbm/basic.py:1286:
UserWarning: Overriding the parameters from Reference Dataset.
  warnings.warn('Overriding the parameters from Reference Dataset.')
/home/azapala/.local/lib/python3.8/site-packages/lightgbm/basic.py:1098:
UserWarning: categorical_column in param dict is overridden.
  warnings.warn('{} in param dict is overridden.'.format(cat_alias))

[1]     train's auc: 0.888823   test's auc: 0.854347
[2]     train's auc: 0.890638   test's auc: 0.855877
[3]     train's auc: 0.894147   test's auc: 0.858108
[4]     train's auc: 0.894462   test's auc: 0.857853
[5]     train's auc: 0.896197   test's auc: 0.855367
[6]     train's auc: 0.898889   test's auc: 0.859128
[7]     train's auc: 0.899832   test's auc: 0.859128
[8]     train's auc: 0.900805   test's auc: 0.858873
[9]     train's auc: 0.903836   test's auc: 0.866331
[10]    train's auc: 0.906427   test's auc: 0.869901
[11]    train's auc: 0.908356   test's auc: 0.874108
[12]    train's auc: 0.908901   test's auc: 0.874363
[13]    train's auc: 0.909672   test's auc: 0.875
[14]    train's auc: 0.911492   test's auc: 0.874426
[15]    train's auc: 0.913362   test's auc: 0.876466
[16]    train's auc: 0.913953   test's auc: 0.873853
[17]    train's auc: 0.914523   test's auc: 0.873534
[18]    train's auc: 0.915278   test's auc: 0.874235
[19]    train's auc: 0.915974   test's auc: 0.876657
[20]    train's auc: 0.920665   test's auc: 0.880801
[21]    train's auc: 0.923323   test's auc: 0.880801
[22]    train's auc: 0.9257     test's auc: 0.880992
[23]    train's auc: 0.926497   test's auc: 0.881502
[24]    train's auc: 0.928136   test's auc: 0.880737
[25]    train's auc: 0.929482   test's auc: 0.88284
[26]    train's auc: 0.930962   test's auc: 0.882458
[27]    train's auc: 0.932995   test's auc: 0.882458
[28]    train's auc: 0.934873   test's auc: 0.877677
[29]    train's auc: 0.935783   test's auc: 0.877868
[30]    train's auc: 0.937854   test's auc: 0.878378
```




```
AUC: 0.8783783783783784
```

Model osiąga relatywnie wysoką metrykę auc na zbiorze testowym. Widzimy równomierny wzrost metryki auc na zbiorze treningowym i testowym. Model się nie przetrenowuje i osiąga zadawalający wynik.

0.10 HP tuning

Przeszukam automatycznie zbiór hyperparametrów i sprawdzę czy tuning pomaga osiągnąć lepsze rezultaty.

[27]:
```
d_train = generate_lgb_dataset(df_train, model_columns, 'Survived') # different␣
↪hps change inner option of dataset, we can't reuse datasets for training
hp_tuner = optuna.integration.lightgbm.LightGBMTunerCV(params, d_train,␣
↪num_boost_round=30, study=optuna.
↪create_study(study_name='automatic-fine-tuning', direction='maximize'),␣
↪verbose_eval=0, return_cvbooster=True)
```

```
[I 2021-03-11 19:29:59,429] A new study created in memory with name:
automatic-fine-tuning
```

[ ]:
```
hp_tuner.run()
```

[29]:
```
hp_tuner.best_score
```

[29]:
```
0.8671699683571635
```

Po autmatycznym przeszukaniu hyperparametrow otrzymaliśmy wyższą metrykę średniego auc na 5-krotnej walidacji krzyżowej: **0.86716**

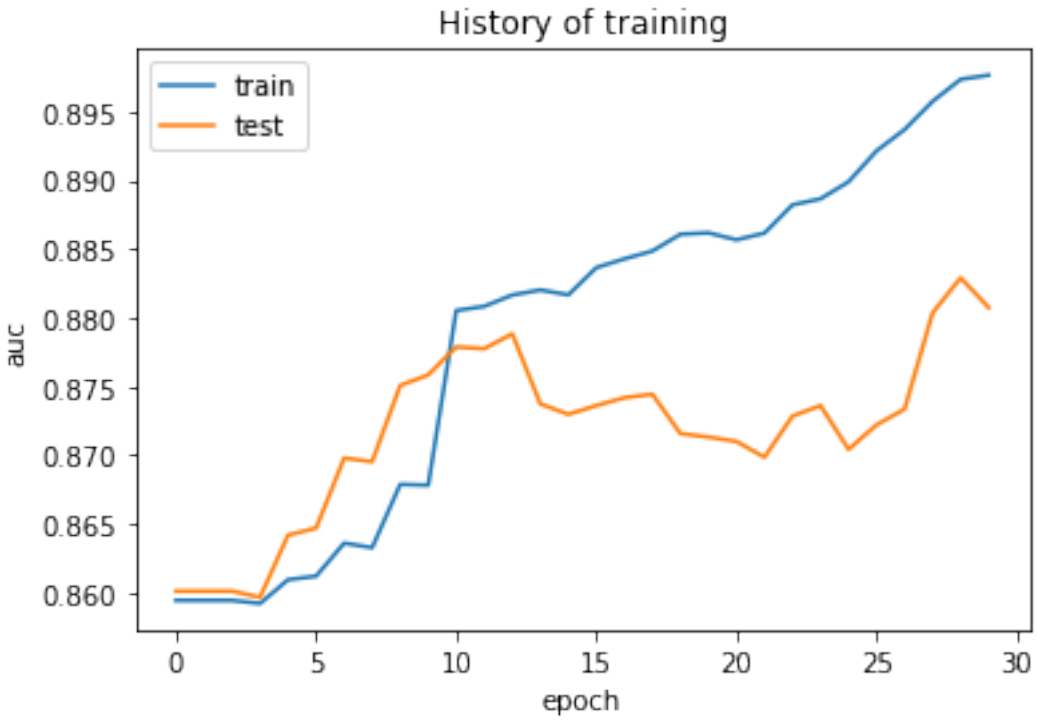
Ten wynik może świadczyć o dopasowaniu hyperparametrów do zbioru treningowego, stąd chcemy upewnić się czy również na zbiorze testowym osiągniemy lepszy wynik.

[30]:
```
params = hp_tuner.best_params
```

[31]:
```
d_train = generate_lgb_dataset(df_train, model_columns, 'Survived')
d_test = generate_lgb_dataset(df_test, model_columns, 'Survived')
train_and_evaluate(params, d_train, d_test)
```

```
/home/azapala/.local/lib/python3.8/site-packages/lightgbm/basic.py:1286:
UserWarning: Overriding the parameters from Reference Dataset.
  warnings.warn('Overriding the parameters from Reference Dataset.')
/home/azapala/.local/lib/python3.8/site-packages/lightgbm/basic.py:1098:
UserWarning: categorical_column in param dict is overridden.
  warnings.warn('{} in param dict is overridden.'.format(cat_alias))
[1]     train's auc: 0.859412   test's auc: 0.860084
[2]     train's auc: 0.859412   test's auc: 0.860084
[3]     train's auc: 0.859412   test's auc: 0.860084
[4]     train's auc: 0.859186   test's auc: 0.859638
[5]     train's auc: 0.860922   test's auc: 0.864164
[6]     train's auc: 0.861173   test's auc: 0.864674
[7]     train's auc: 0.86358    test's auc: 0.869773
[8]     train's auc: 0.863253   test's auc: 0.869518
[9]     train's auc: 0.867856   test's auc: 0.875064
[10]    train's auc: 0.867797   test's auc: 0.875829
[11]    train's auc: 0.880522   test's auc: 0.877868
[12]    train's auc: 0.880823   test's auc: 0.877741
[13]    train's auc: 0.881649   test's auc: 0.878825
[14]    train's auc: 0.882018   test's auc: 0.873725
[15]    train's auc: 0.881662   test's auc: 0.87296
[16]    train's auc: 0.883645   test's auc: 0.873598
[17]    train's auc: 0.884282   test's auc: 0.874171
[18]    train's auc: 0.884852   test's auc: 0.874426
[19]    train's auc: 0.886077   test's auc: 0.871558
[20]    train's auc: 0.886177   test's auc: 0.871303
[21]    train's auc: 0.88567    test's auc: 0.870984
[22]    train's auc: 0.886156   test's auc: 0.869837
[23]    train's auc: 0.888219   test's auc: 0.872833
[24]    train's auc: 0.888655   test's auc: 0.873598
[25]    train's auc: 0.889896   test's auc: 0.870411
[26]    train's auc: 0.892152   test's auc: 0.872195
[27]    train's auc: 0.893703   test's auc: 0.873343
[28]    train's auc: 0.895757   test's auc: 0.880354
```

```
[29]    train's auc: 0.897359   test's auc: 0.882904
[30]    train's auc: 0.897661   test's auc: 0.880737
```




```
AUC: 0.8807368689444162
```

Na zbiorze testowym również poprawiliśmy metrykę auc: **0.8807** w porównaniu do *0.8783*, stąd możemy wysunąć wniosek, że automatyczne dopasowanie hyperparametrów poprawiło model.