Тема: XGBoost за класификация на доход

Изготвил: Кирил Романски

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype 
---  ------          --------------  ----- 
 0   age             32561 non-null  int64 
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64 
 3   education       32561 non-null  object
 4   marital_status  32561 non-null  int64 
 5   occupation      32561 non-null  object
 6   specialty       32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital-gain    32561 non-null  int64 
 11  capital-loss    32561 non-null  int64 
 12  hours_per_week  32561 non-null  int64 
 13  native_country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

Данни:

- 15 променливи: age, workclass, fnlwgt,education,marital-status, occupation,relationship, race, sex, capital-gain, capital-loss, hours-per-week, native-coutnry

Допълнителен файл с описание:

```
>50K, <=50K.

age: continuous.
workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.
fnlwgt: continuous.
education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters,
1st-4th, 10th, Doctorate, 5th-6th, Preschool.
education-num: continuous.
marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-
spouse.
occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners,
Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.
relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.
race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.
sex: Female, Male.
capital-gain: continuous.
capital-loss: continuous.
hours-per-week: continuous.
native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India,
Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal,
Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland,
Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.
```

Обработка на данни 1:

- Тъй като ще използваме XGboost алгоритъм ще трябва да превърнем всички данни които не са числови в числови със подходящо кодиране на променливите:
- Maritual_status- Тази е компрометирана в дейта сетта. Първо е кодирана и второ вместо с 7 категории, които има в описанието на данните в предишния слайд, са използвани числата от 1-13 за кодиране. Променливата бе премахната от решението.
- Education- Кодирана 1-13 за всяка категория. Maritual_status в оригиналния дейтасет всъщност отговаряше на това кодиране. Тази променлива е кодирана по този начин заради натуралния смисъл на ред в образованието.

Обработка на данни 2

- Race&Sex-Кодирани с True и False.
- Income-Целева променлива кодирана с 0 за доход >=50k, 1 за доход <50k.
- Workclass, Specialty, Occupation,Relationship- Кодирани пак с True и False, но преди това поради наличие на празни записи-"?" Трансформация "?"->"Unknown"
- Native_Country-Тази колона има 45 различни стойности като от 32000 записа ~29000 са United_states и още 6 имат повече от 100 записа за това съм ги групирал на 7 променливи- 6 за стани с повече от 100 наблюдния и останалите <100 като сборна променлива "Rare_countries"

За метода: XGBoost

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$$

$$\text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \|w\|^2$$

- Основна идея: Метода е на стъпки. Всяка итерация добавя класификационно дърво, което намалява грешката с градиентно оптимизиране. Разликата с Gradient Boosting-а е регуляризацията, която се добавя като мярка срещу overfit.
- Минимизира се с ред на Тейлър понеже базовата функция е много трудна за диференциране
- l-функция на грешката и когато става въпрос за класификация използваме логистична загуба

Още за метода:

- Функция на загубата(в нашия случай с бинарна класификация):

$$\mathcal{L}(y, \hat{y}) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

- Негативен log likelihood
- Тук y-реална стойност y(bar)- вероятност! Предсказана от модела.

Важни концепции за XGboost

- Greedy Algorithm- Approximate Greedy Algorithm
- Parallel Learning
- Sparsity-Aware Split Finding
- Cache-Aware Access

Хиперпараметри

- Ще оптимизираме следните хиперпараметри на алгоритъма:
- Максимална дълбочина на дърветата
- Gamma- долна граница за Gain
- Learning rate- стъпка
- Брой дървета
- min_child_weight

Резултат от начално изпълнение на алгоритъма:

```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:}")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8545984953170582
              precision    recall  f1-score   support

           0       0.88      0.93      0.91      4942
           1       0.74      0.62      0.67      1571

    accuracy                           0.85      6513
   macro avg       0.81      0.77      0.79      6513
weighted avg       0.85      0.85      0.85      6513
```

Резултат след крос валидация:

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
cv_scores = cross_val_score(model, X, y, cv=skf, scoring='accuracy')
print(f'F1 scores for each fold: {cv_scores}')
print(f'Average F1 score: {cv_scores.mean():}')
```

```
F1 scores for each fold: [0.7968678  0.7919226  0.79376536 0.79422604 0.79960074]
Average F1 score: 0.7952765089741138
```

StratifiedKFold вместо обикновен K-fold понеже разпределението на целевия ни атрибут не е равномерно тоест единия клас доминира. StratifiedKFold се грижи разпределението да остане приблизително еднакво на раделенията на групите.

```
income
<=50K    24720
>50K      7841
Name: count, dtype: int64
```

Тестване на хипер параметри: Grid-Search

```python
param_grid = {
    'max_depth': [3, 5, 7],
    'min_child_weight': [1, 3, 5],
    'gamma': [0, 0.1, 0.2],
    'subsample': [0.8, 1],
    'colsample_bytree': [0.8, 1],
    'learning_rate': [0.01, 0.1],
    'n_estimators': [100, 200]
}
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
grid_search = GridSearchCV(
    estimator=model,
    param_grid=param_grid,
    scoring='accuracy',
    cv=skf,
    verbose=2,
    n_jobs=-1
)
grid_search.fit(X, y)
print("Best parameters found: ", grid_search.best_params_)
print(f"Best F1 score: {grid_search.best_score_:}")
```

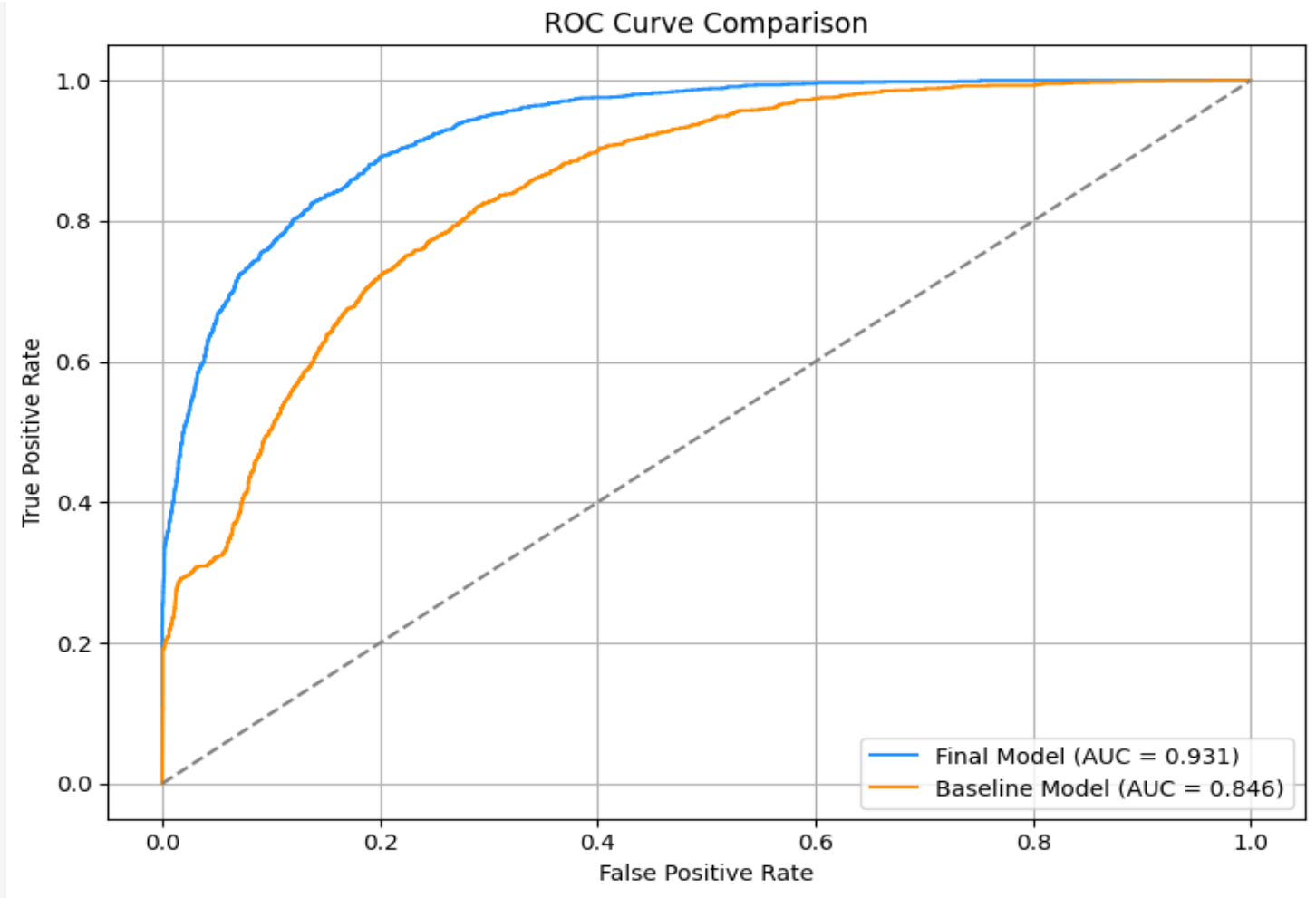
```
Fitting 5 folds for each of 432 candidates, totalling 2160 fits
Best parameters found:  {'colsample_bytree': 0.8, 'gamma': 0.1, 'learning_rate': 0.1, 'max_depth': 5, 'min_child_weight': 1, 'n_estimators': 200, 'subsa
mple': 0.8}
Best F1 score: 0.8748
```

Финален модел с най-добрите параметри

```
y_pred_final = final_model.predict(X_test)

print(f"Accuracy: {accuracy_score(y_test, y_pred):}")
print(f"F1 Score: {f1_score(y_test, y_pred_final):}")
print(classification_report(y_test, y_pred_final))
```

```
Accuracy: 0.8545984953170582
F1 Score: 0.7319904404233527
              precision    recall  f1-score   support

           0       0.90      0.94      0.92      4942
           1       0.79      0.68      0.73      1571

    accuracy                           0.88      6513
   macro avg       0.85      0.81      0.83      6513
weighted avg       0.88      0.88      0.88      6513
```

ROC curve сравнение: финален срещу базов



Резултат от регресия:

Logit Regression Results

Dep. Variable:	income	No. Observations:	32561
Model:	Logit	Df Residuals:	32518
Method:	MLE	Df Model:	42
Date:	Fri, 13 Jun 2025	Pseudo R-squ.:	0.4238
Time:	12:52:43	Log-Likelihood:	-10357.
converged:	False	LL-Null:	-17974.
Covariance Type:	nonrobust	LLR p-value:	0.000

Използвах и крос валидация отново и резултата не е лош в интерес на истината Cross-validated Accuracy scores: [0.83816981 0.83860565 0.83522727 0.8379914 0.84336609]
Average Accuracy: 0.8387

KNN-най-близки съседи

- Предварителни обработки: Стандартизиране на всички променливи, понеже работим с векторни разстояния.
- Резултат:

KNN Classifier
Accuracy: 0.8300

	precision	recall	f1-score	support
0	0.87	0.92	0.89	4942
1	0.68	0.56	0.61	1571
accuracy			0.83	6513
macro avg	0.77	0.74	0.75	6513
weighted avg	0.82	0.83	0.82	6513

- Интерпретация: Слабо класифициране на класа >50k

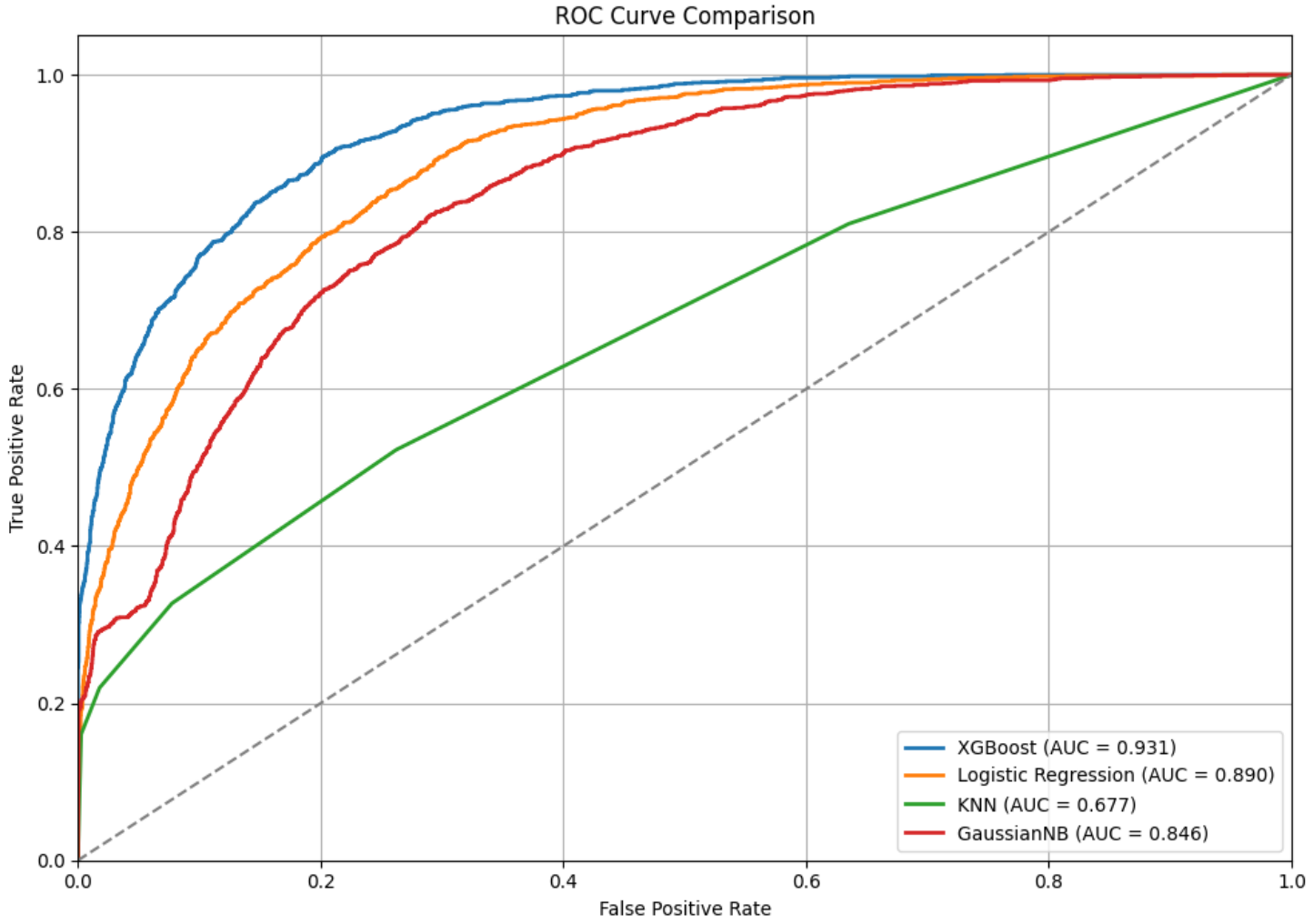
Найвен Бейсов класификатор

- Резултат:

```
Gaussian Naive Bayes
Accuracy: 0.5441
              precision    recall  f1-score   support

           0       0.97      0.41      0.58      4942
           1       0.34      0.96      0.50      1571

    accuracy                           0.54      6513
   macro avg       0.65      0.68      0.54      6513
weighted avg       0.82      0.54      0.56      6513
```

- Интерпретация: Добре разпознава доминирация клас, нищо друго.

Графично сравнение на алгоритмите:



Източници:

- XGBoost Documentation — xgboost 3.1.0-dev documentation
- ISLP