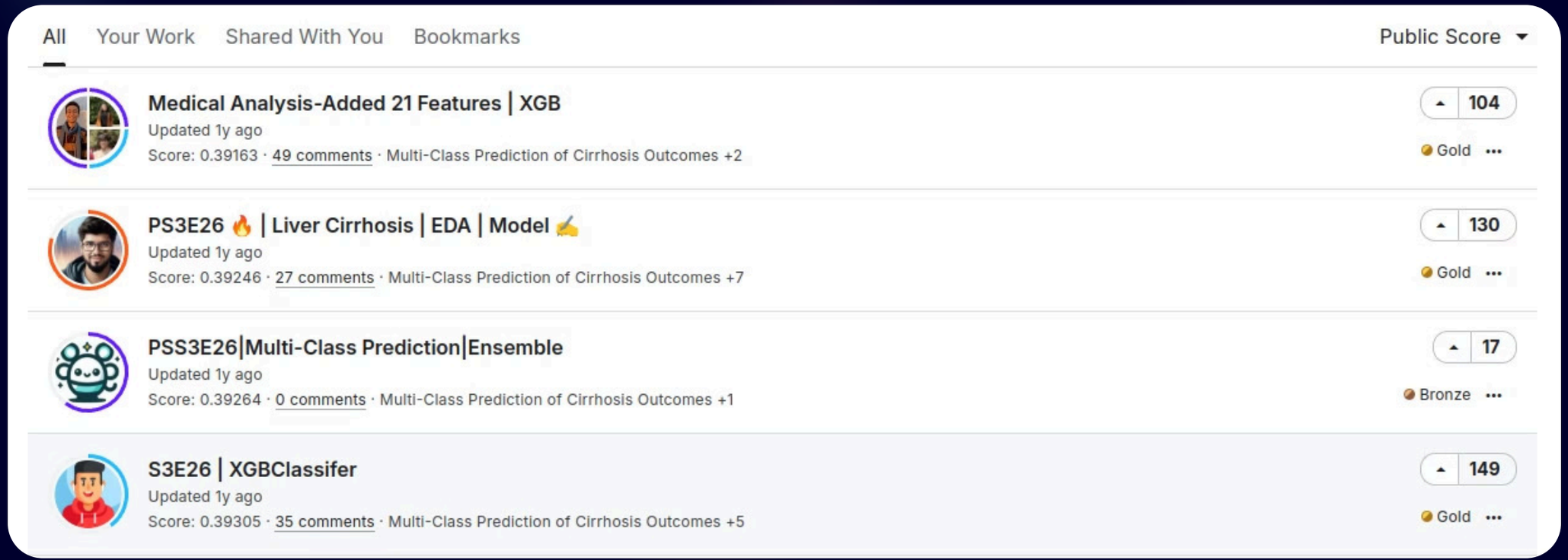
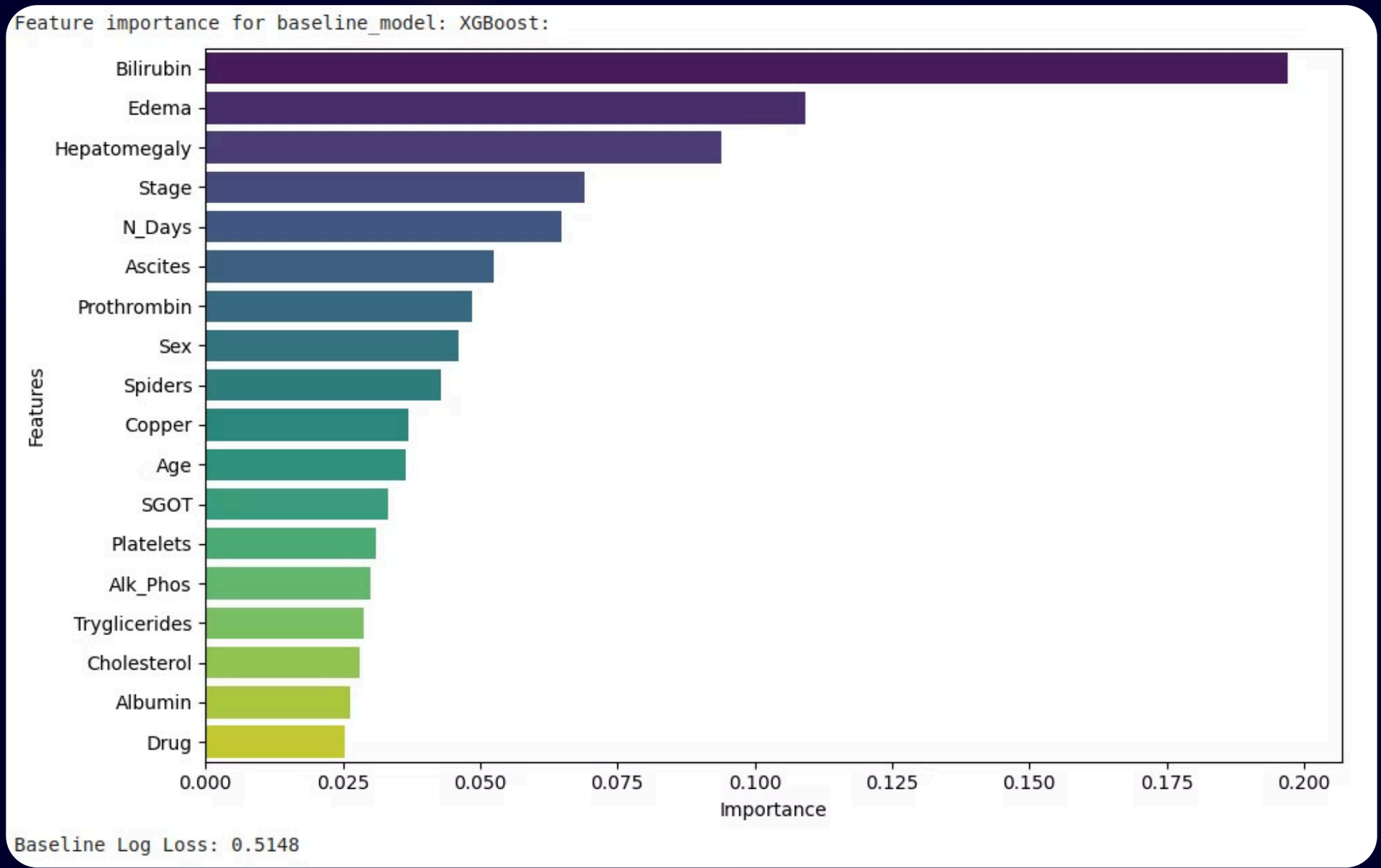
Multi-Class Prediction of Cirrhosis Outcomes

PUBLIC: 0.45518



Baseline XGBoost

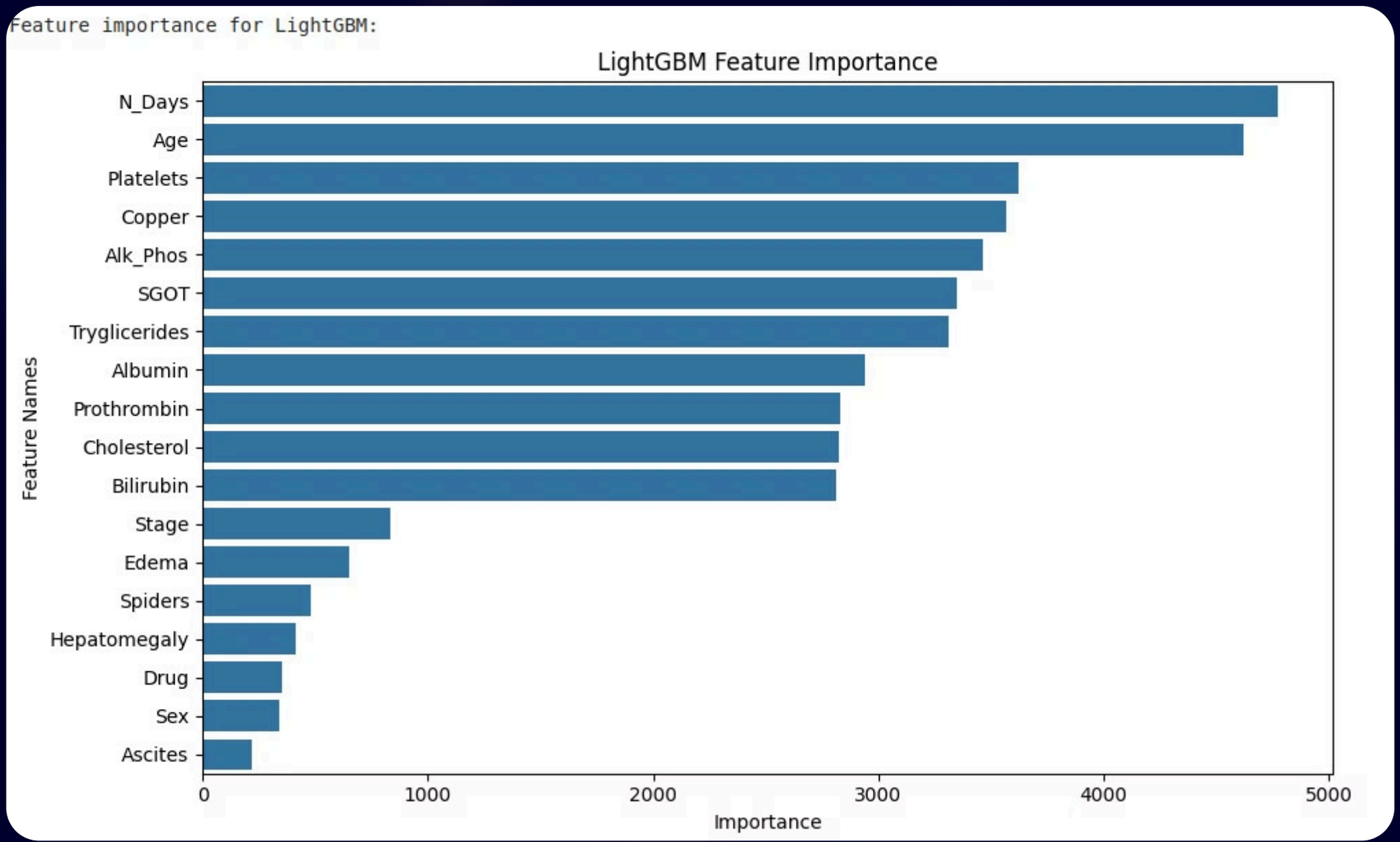
Baseline XGBoostLog Loss: 0.5148



LightGBM

ldbm Log Loss: 0.4445

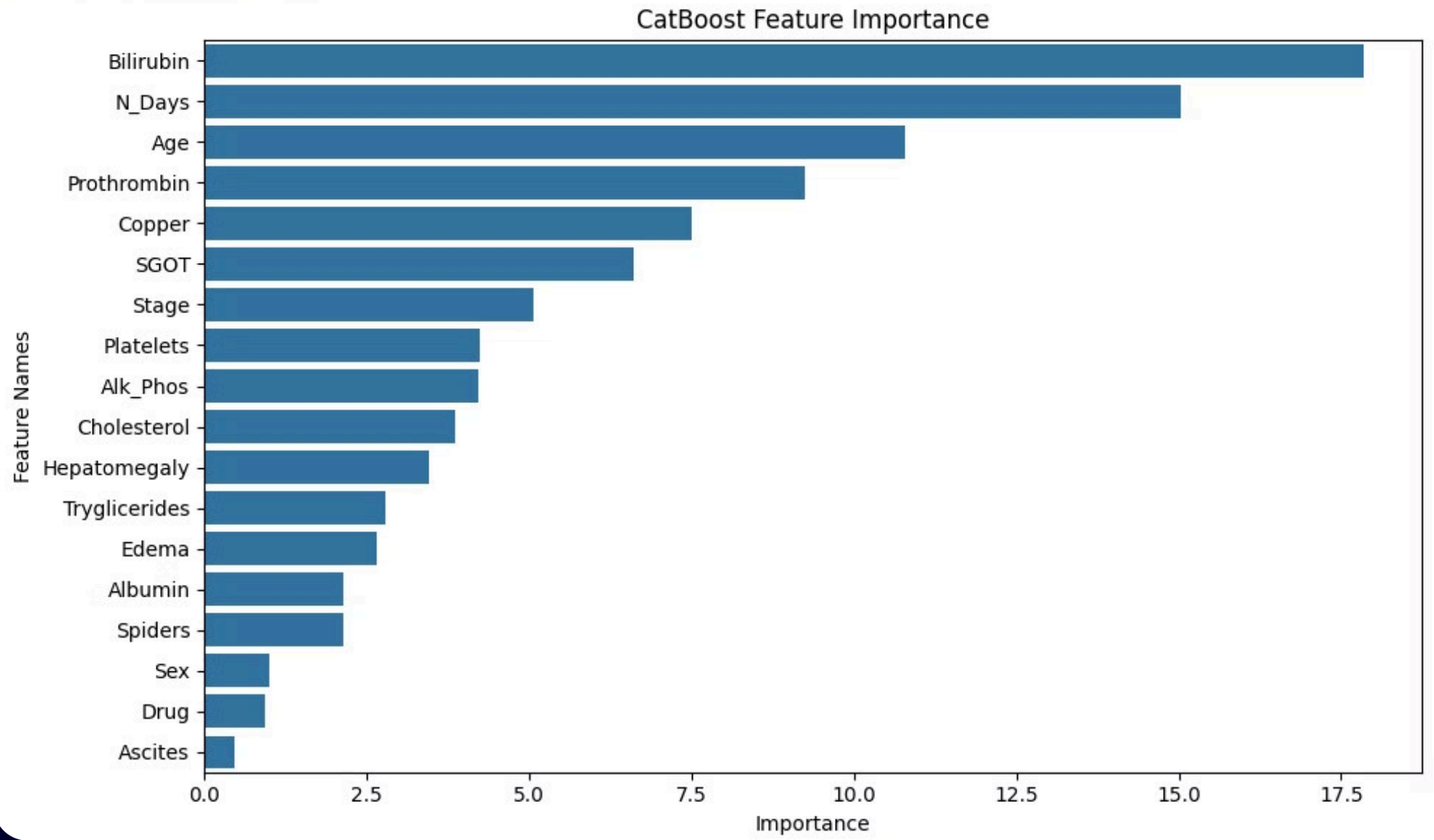
Feature importance for LightGBM:



CatBoost

catboost Log Loss: 0.4757

Feature importance for CatBoost:



LightGBM && CatBoost Ensemble

Ensemble Log Loss: 0.4499

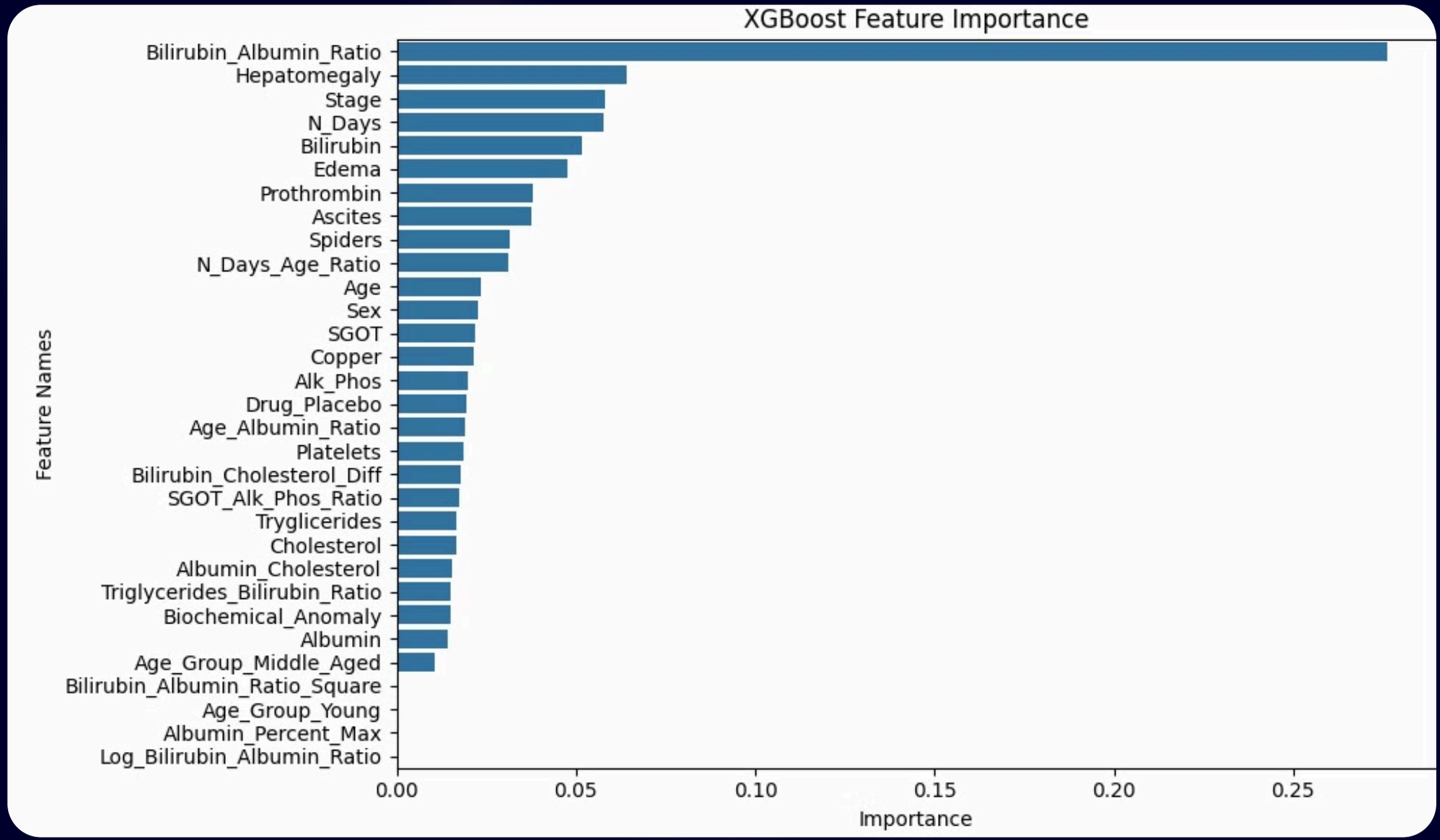
```
ensemble_clear = VotingClassifier(
    estimators=[
        ('lgbm', lgbm),
        ('catboost', catboost)
    ],
    voting='soft',
    weights=[1, 1]
)

ensemble_clear.fit(X_train, y_train_encoded)

ensemble_proba = ensemble_clear.predict_proba(X_val)
ensemble_proba = np.clip(ensemble_proba, 1e-15, 1-1e-15)

ensemble_clear_logloss = log_loss(y_val_encoded, ensemble_proba)
print(f"Ensemble Log Loss: {ensemble_clear_logloss:.4f}")
```

Генерация новых признаков для улучшения моделей

```python
# One-Hot encoding of age groups
train_data_processed = pd.get_dummies(train_data_processed, columns=['Age_Group'], drop_first=True)
test_data_processed = pd.get_dummies(test_data_processed, columns=['Age_Group'], drop_first=True)

# Generating ratios between indicators
train_data_processed['Bilirubin_Albumin_Ratio'] = train_data_processed['Bilirubin'] / train_data_processed['Albumin']
train_data_processed['SGOT_Alk_Phos_Ratio'] = train_data_processed['SGOT'] / train_data_processed['Alk_Phos']

test_data_processed['Bilirubin_Albumin_Ratio'] = test_data_processed['Bilirubin'] / test_data_processed['Albumin']
test_data_processed['SGOT_Alk_Phos_Ratio'] = test_data_processed['SGOT'] / test_data_processed['Alk_Phos']

train_data_processed['Bilirubin_Albumin_Ratio_Square'] = train_data_processed['Bilirubin_Albumin_Ratio'] ** 2
test_data_processed['Bilirubin_Albumin_Ratio_Square'] = test_data_processed['Bilirubin_Albumin_Ratio'] ** 2

# Logarithmic transformation of Bilirubin
train_data_processed['Log_Bilirubin_Albumin_Ratio'] = train_data_processed['Bilirubin_Albumin_Ratio'].apply(lambda x: np.log(x) if x > 0 else 0)
test_data_processed['Log_Bilirubin_Albumin_Ratio'] = test_data_processed['Bilirubin_Albumin_Ratio'].apply(lambda x: np.log(x) if x > 0 else 0)

# Normalizing days by age
train_data_processed['N_Days_Age_Ratio'] = train_data_processed['N_Days'] / train_data_processed['Age']
test_data_processed['N_Days_Age_Ratio'] = test_data_processed['N_Days'] / test_data_processed['Age']

# Polynomial features
train_data_processed['Albumin_Cholesterol'] = train_data_processed['Albumin'] * train_data_processed['Cholesterol']
test_data_processed['Albumin_Cholesterol'] = test_data_processed['Albumin'] * test_data_processed['Cholesterol']

# Feature: Bilirubin deviation from median
median_bilirubin_train = train_data_processed['Bilirubin'].median()
median_bilirubin_test = test_data_processed['Bilirubin'].median()

# Albumin percentage of maximum observed
max_albumin_train = train_data_processed['Albumin'].max()
max_albumin_test = test_data_processed['Albumin'].max()

train_data_processed['Albumin_Percent_Max'] = train_data_processed['Albumin'] / max_albumin_train
test_data_processed['Albumin_Percent_Max'] = test_data_processed['Albumin'] / max_albumin_test
```

XGBoost после генерации

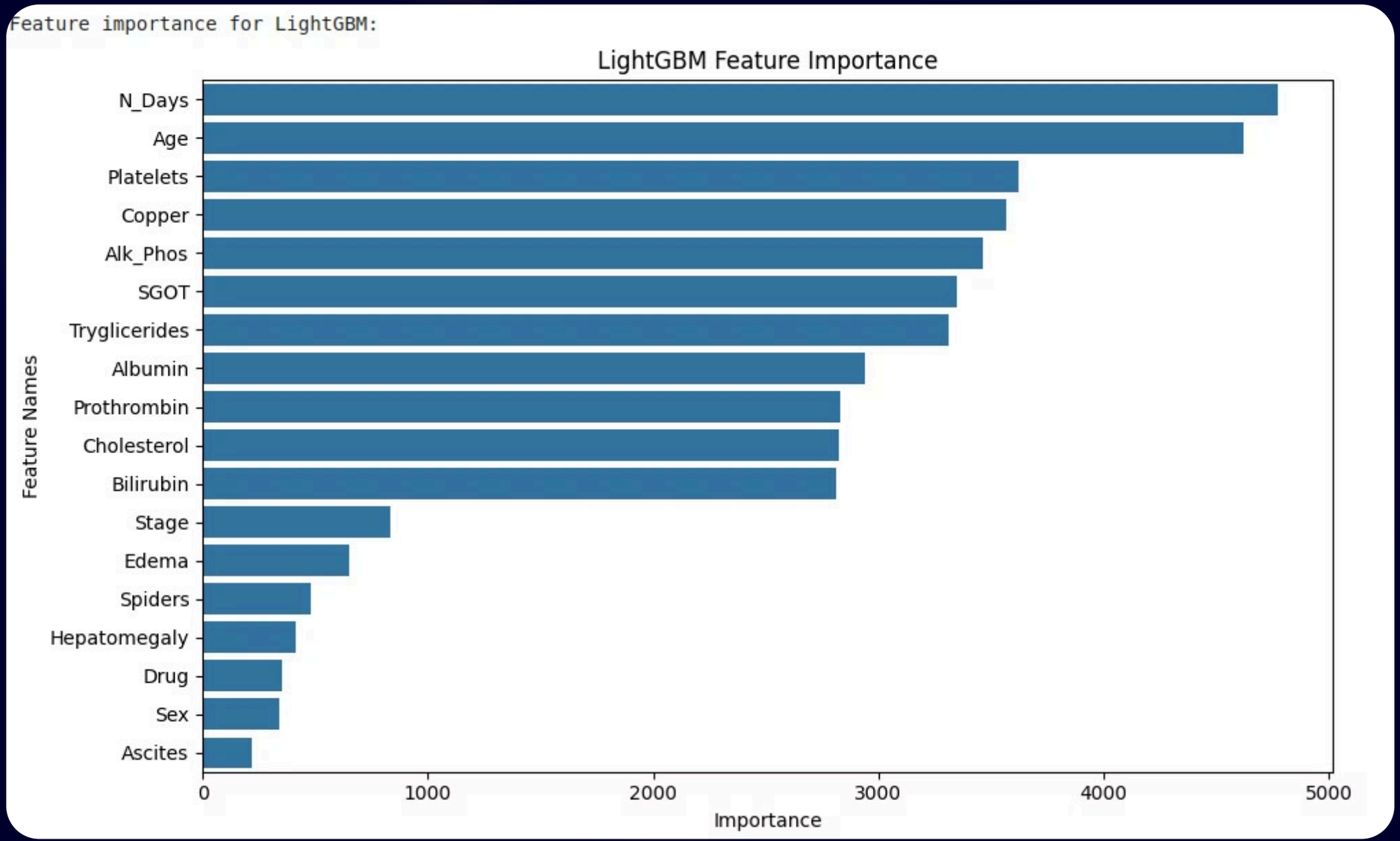
xgb Log Loss: 0.4597



LightGBM после генерации

ldbm Log Loss: 0.4491

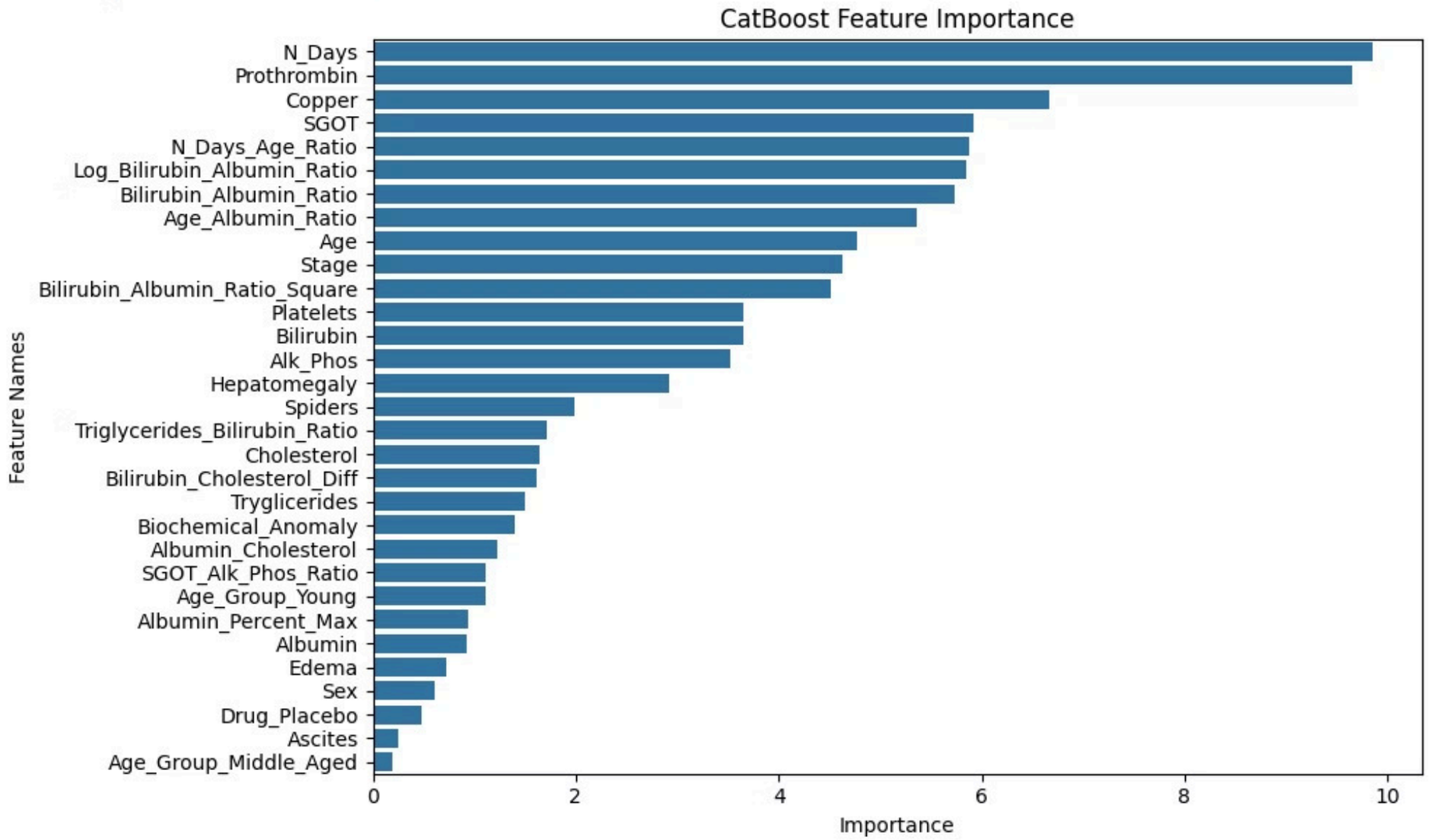
Feature importance for LightGBM:



CatBoost после генерации

catboost Log Loss: 0.4783

Feature importance for CatBoost:



XGBoost && LightGBM && CatBoost Ensemble

Ensemble Log Loss: 0.4531

```python
# Ensemble
feature_gen_ensemble = VotingClassifier(
    estimators=[
        ('xgb', xgb),
        ('lgbm', lgbm),
        ('catboost', catboost)
    ],
    voting='soft',
    weights=[1, 1, 1]
)
feature_gen_ensemble.fit(X_train, y_train_encoded)

ensemble_proba = feature_gen_ensemble.predict_proba(X_val)
ensemble_proba = np.clip(ensemble_proba, 1e-15, 1-1e-15)

feature_gen_ensemble_logloss = log_loss(y_val_encoded, ensemble_proba)
print(f"Ensemble Log Loss: {feature_gen_ensemble_logloss:.4f}")
```

```
xgb Log Loss: 0.4597
ldbm Log Loss: 0.4491
catboost Log Loss: 0.4783
```

Улучшенная предобработка данных

1

Обработка пропусков

Числовые: медиана. Категориальные: мода.

2

Удаление выбросов

IQR для обрезки экстремальных значений.

3

Кодирование категориальных признаков

LabelEncoder для преобразования в числовой формат.

4

Масштабирование и трансформация

PowerTransformer для нормализации числовых признаков.

5

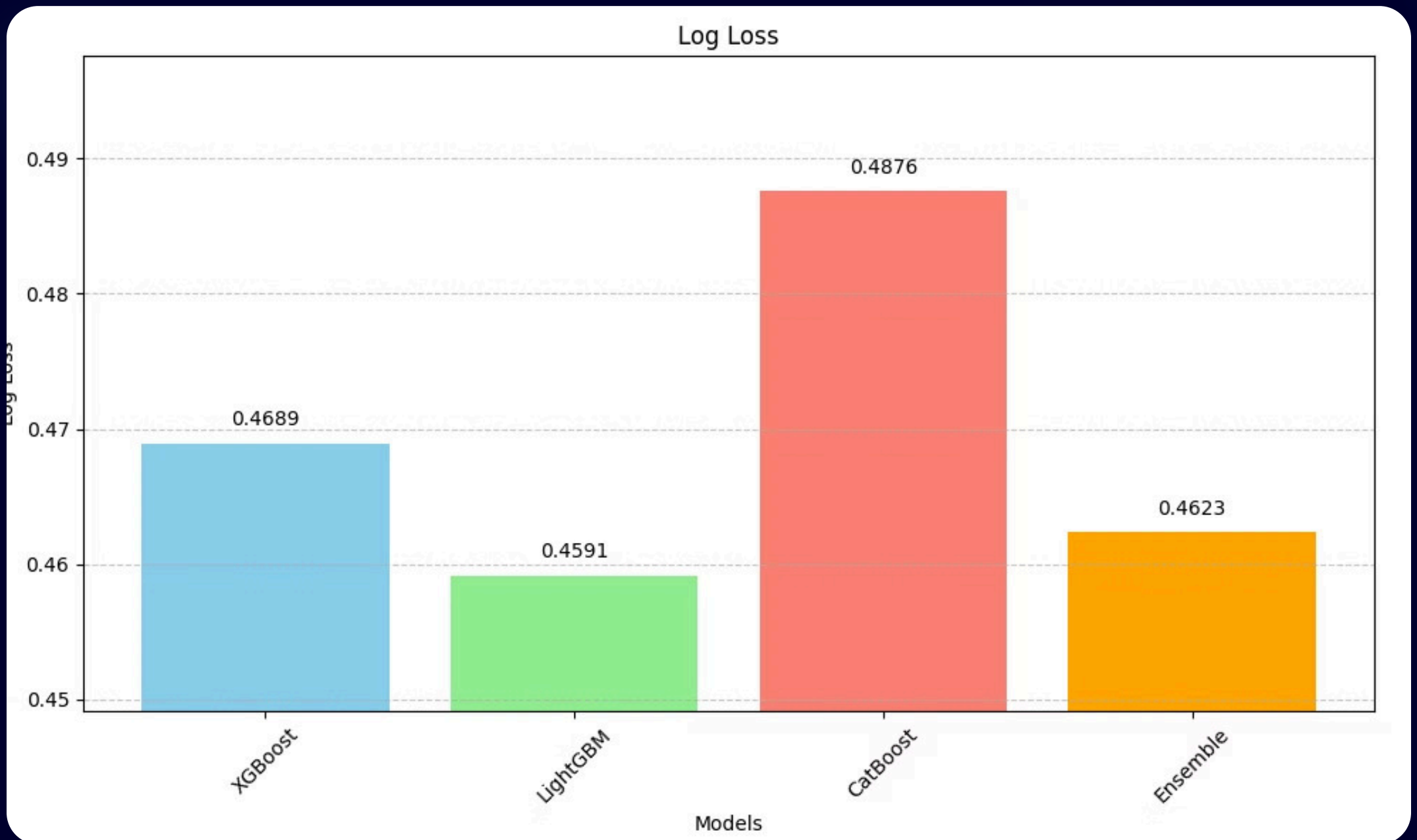
Добавление шума

Шум добавлен с использованием add_noise.

Feature Jittering

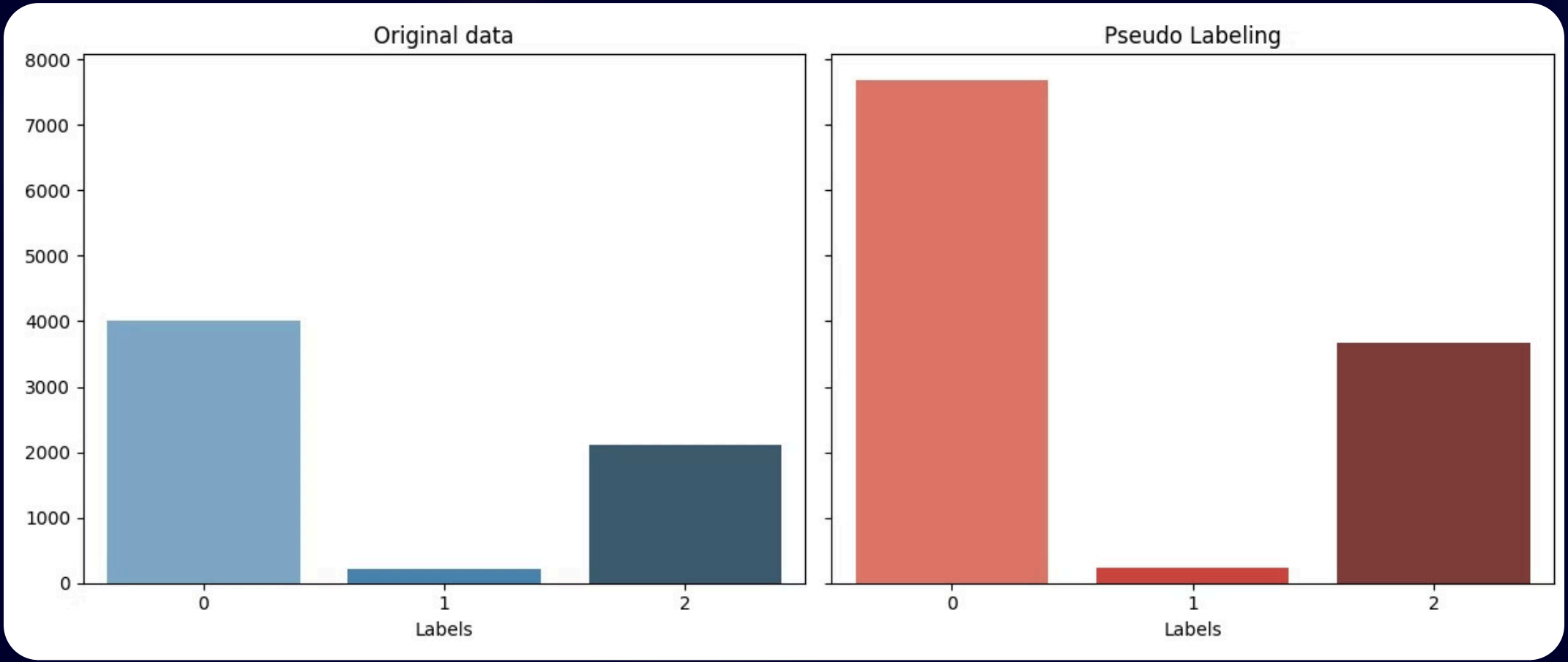
Ensemble Log Loss: 0.4623

```python
def add_noise(df, noise_level=0.01):
    numeric_cols = df.select_dtypes(include=['float64', 'int64']).columns
    noisy_df = df.copy()
    for col in numeric_cols:
        noise = np.random.normal(0, noise_level * df[col].std(), size=len(df))
        noisy_df[col] = df[col] + noise
    return noisy_df
```



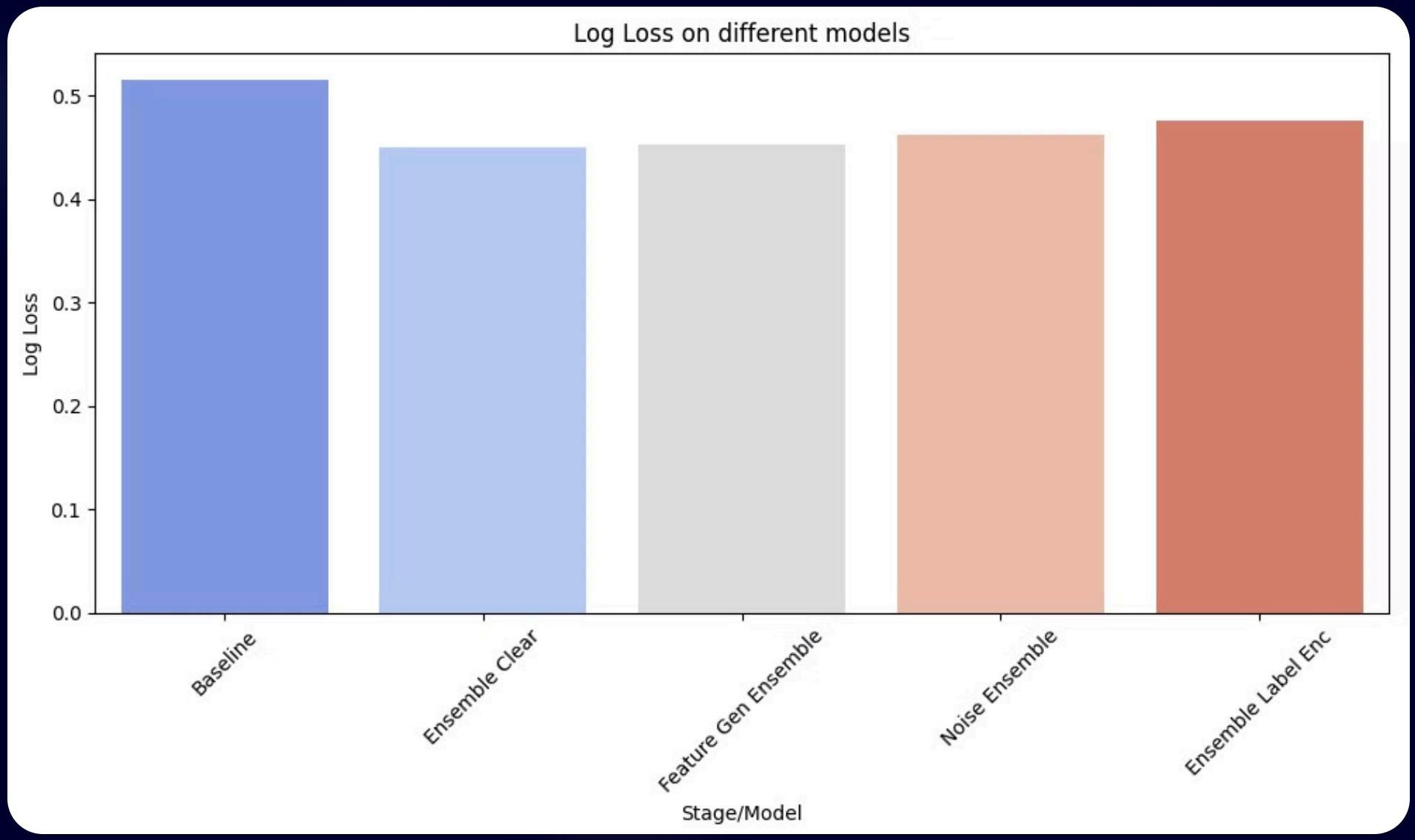
Pseudo labeling

Ensemble Log Loss with Pseudo-labeling: 0.4756



Сравнение моделей

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij}),$$



0.4623 0.4756 0.5148 0.4499 0.4531

Основные Выводы

Ансамблевые Модели

Значительное снижение Log Loss по сравнению с базовой моделью.

Ensemble Clear

Продемонстрировал наилучшие результаты, став основной моделью

Генерация Признаков, Feature Jittering и Pseudo-labeling

Дополнительный потенциал для улучшения качества моделей.