In [1]:
```python
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.impute import SimpleImputer
from sklearn.metrics import accuracy_score, classification_report

# Загрузка данных
df = pd.read_csv("reduced.csv")

# Удалим ненужные признаки
df = df.drop(columns=["url", "status", "status_encoded"])

# Отделим целевую переменную
X = df.drop("is_malicious", axis=1)
y = df["is_malicious"]

# Категориальные и числовые признаки
categorical = ["tld", "extension"]
numerical = [col for col in X.columns if col not in categorical]

# Преобразователь
preprocessor = ColumnTransformer([
    ("num", SimpleImputer(strategy="mean"), numerical),
    ("cat", Pipeline(steps=[
    ("imputer", SimpleImputer(strategy="most_frequent")),
    ("encoder", OneHotEncoder(handle_unknown="ignore", sparse_output=False))
]), categorical)

])

# Разделение на train/test и сброс индексов
X_train, X_test, y_train, y_test = train_test_split(
    X.reset_index(drop=True),
    y.reset_index(drop=True),
    test_size=0.2,
    random_state=42
)

# Трансформация признаков
X_train_transformed = preprocessor.fit_transform(X_train)
X_test_transformed = preprocessor.transform(X_test)

# Преобразуем в DataFrame с правильными именами признаков
X_train_transformed_df = pd.DataFrame(X_train_transformed, columns=preproces

# Проверим пропуски
print("NaN в train:", np.isnan(X_train_transformed).sum())
print("NaN в test:", np.isnan(X_test_transformed).sum())

print(X_train_transformed_df.head())
```

```
NaN в train: 0
NaN в test: 0
   num__url_length  num__tls  num__special_chars_count  cat__tld_10  \
0        -0.447108       1.0                 -0.341804          0.0
1        -0.429314       0.0                 -0.375883          0.0
2        -0.180196       0.0                 -0.205490          0.0
3        -0.500491       1.0                 -0.307726          0.0
4         0.140099       0.0                 -0.001019          0.0

   cat__tld_12  cat__tld_162  cat__tld_166  cat__tld_185  cat__tld_186  \
0          0.0           0.0           0.0           0.0           0.0
1          0.0           0.0           0.0           0.0           0.0
2          0.0           0.0           0.0           0.0           0.0
3          0.0           0.0           0.0           0.0           0.0
4          0.0           0.0           0.0           0.0           0.0

   cat__tld_195  ...  cat__extension_.site  cat__extension_.sk  \
0           0.0  ...                   0.0                 0.0
1           0.0  ...                   0.0                 0.0
2           0.0  ...                   0.0                 0.0
3           0.0  ...                   0.0                 0.0
4           0.0  ...                   0.0                 0.0

   cat__extension_.su  cat__extension_.top  cat__extension_.txt  \
0                 0.0                  0.0                  0.0
1                 0.0                  0.0                  0.0
2                 0.0                  0.0                  0.0
3                 0.0                  0.0                  0.0
4                 0.0                  0.0                  0.0

   cat__extension_.ua  cat__extension_.uk  cat__extension_.vn  \
0                 0.0                 0.0                 0.0
1                 0.0                 0.0                 0.0
2                 0.0                 0.0                 0.0
3                 0.0                 0.0                 0.0
4                 0.0                 0.0                 0.0

   cat__extension_.za  cat__extension_none
0                 0.0                  1.0
1                 0.0                  0.0
2                 0.0                  0.0
3                 0.0                  1.0
4                 0.0                  0.0

[5 rows x 216 columns]
```

In [2]:
```python
# Обучение моделей
```

In [3]:
```python
# Стекинг
print("Обучение...")
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassif

base_learners = [
    ("rf", RandomForestClassifier(n_estimators=50, random_state=42)),
    ("gb", GradientBoostingClassifier(n_estimators=50, random_state=42))
```

```
]

stack_model = Pipeline([
    ("pre", preprocessor),
    ("clf", StackingClassifier(
        estimators=base_learners,
        final_estimator=LogisticRegression(),
        cv=5
    ))
])

stack_model.fit(X_train, y_train)
y_pred_stack = stack_model.predict(X_test)
print("OK")
```

```
Обучение...
OK
```

In [4]:

```
from sklearn.neural_network import MLPClassifier
# Многослойный персептрон
print("Обучение...")
mlp_model = Pipeline([
    ("pre", preprocessor),
    ("clf", MLPClassifier(hidden_layer_sizes=(64, 32), max_iter=500, random_
])

mlp_model.fit(X_train, y_train)
y_pred_mlp = mlp_model.predict(X_test)
print("OK")
```

```
Обучение...
OK
```

In [5]:

```
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import PolynomialFeatures
from sklearn.metrics import accuracy_score, classification_report

# === Подготовка полиномиальных признаков ===
poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=Fals
X_train_poly = poly.fit_transform(X_train_transformed)
X_test_poly = poly.transform(X_test_transformed)

# === Линейная модель COMBI (полиномиальный Lasso) ===
combi_model = Lasso(alpha=0.01, max_iter=10000)
combi_model.fit(X_train_poly, y_train)
y_pred_combi_continuous = combi_model.predict(X_test_poly)
# Округляем для классификации (если задача бинарная)
y_pred_combi = (y_pred_combi_continuous > 0.5).astype(int)

combi_acc = accuracy_score(y_test, y_pred_combi)
print("\nЛинейная модель COMBI (полиномиальный Lasso):")
print("Accuracy:", combi_acc)
print(classification_report(y_test, y_pred_combi))

# === Нелинейная модель MIA (полиномиальные признаки + случайный лес) ===
mia_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```python
mia_model.fit(X_train_poly, y_train)
y_pred_mia = mia_model.predict(X_test_poly)

mia_acc = accuracy_score(y_test, y_pred_mia)
print("\nНелинейная модель MIA (полиномиальные признаки + RandomForest):")
print("Accuracy:", mia_acc)
print(classification_report(y_test, y_pred_mia))
```

```
Линейная модель COMBI (полиномиальный Lasso):
Accuracy: 0.6683333333333333
              precision    recall  f1-score   support

           0       0.62      0.82      0.70       288
           1       0.76      0.53      0.63       312

    accuracy                           0.67       600
   macro avg       0.69      0.67      0.66       600
weighted avg       0.69      0.67      0.66       600


Нелинейная модель MIA (полиномиальные признаки + RandomForest):
Accuracy: 0.75
              precision    recall  f1-score   support

           0       0.73      0.77      0.75       288
           1       0.78      0.73      0.75       312

    accuracy                           0.75       600
   macro avg       0.75      0.75      0.75       600
weighted avg       0.75      0.75      0.75       600
```

In [6]:

```python
def evaluate(y_true, y_pred, name):
    print(f"Model: {name}")
    print("Accuracy: {:.2f}".format(accuracy_score(y_true, y_pred)))
    print("Report:\n", classification_report(y_true, y_pred))

evaluate(y_test, y_pred_stack, "Stacking")
evaluate(y_test, y_pred_mlp, "MLP")
evaluate(y_test, y_pred_combi, "COMBI (GMDH)")
evaluate(y_test, y_pred_mia, "MIA (GMDH)")
```

```
Model: Stacking
Accuracy: 0.77
Report:
               precision    recall  f1-score   support

           0       0.72      0.84      0.78       288
           1       0.83      0.70      0.76       312

    accuracy                           0.77       600
   macro avg       0.77      0.77      0.77       600
weighted avg       0.78      0.77      0.77       600

Model: MLP
Accuracy: 0.76
Report:
               precision    recall  f1-score   support

           0       0.71      0.83      0.76       288
           1       0.81      0.69      0.74       312

    accuracy                           0.76       600
   macro avg       0.76      0.76      0.75       600
weighted avg       0.76      0.76      0.75       600

Model: COMBI (GMDH)
Accuracy: 0.67
Report:
               precision    recall  f1-score   support

           0       0.62      0.82      0.70       288
           1       0.76      0.53      0.63       312

    accuracy                           0.67       600
   macro avg       0.69      0.67      0.66       600
weighted avg       0.69      0.67      0.66       600

Model: MIA (GMDH)
Accuracy: 0.75
Report:
               precision    recall  f1-score   support

           0       0.73      0.77      0.75       288
           1       0.78      0.73      0.75       312

    accuracy                           0.75       600
   macro avg       0.75      0.75      0.75       600
weighted avg       0.75      0.75      0.75       600
```

In [ ]: