ИНСТИТУТ ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
УНИВЕРСИТЕТА ИННОПОЛИС

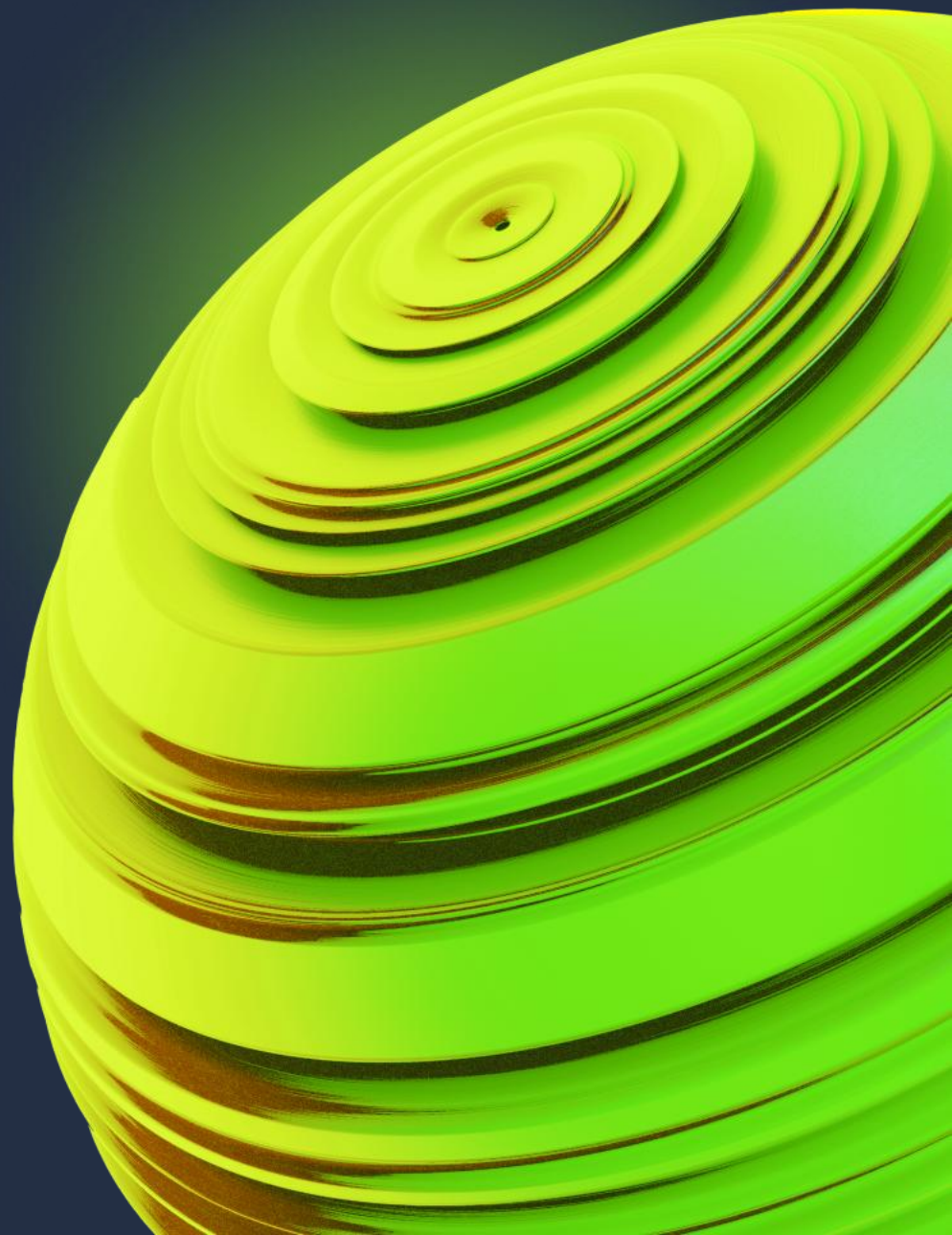
УНИВЕРСИТЕТ ИННОПОЛИС

Программная инженерия: ИТ-лидеры будущего

Презентация к итоговой аттестационной работе

Ильина Ксения Сергеевна

2024



Тема

ИТОГОВАЯ АТТЕСТАЦИОННАЯ РАБОТА ПО КУРСУ
«Программная инженерия: ИТ-лидеры будущего»

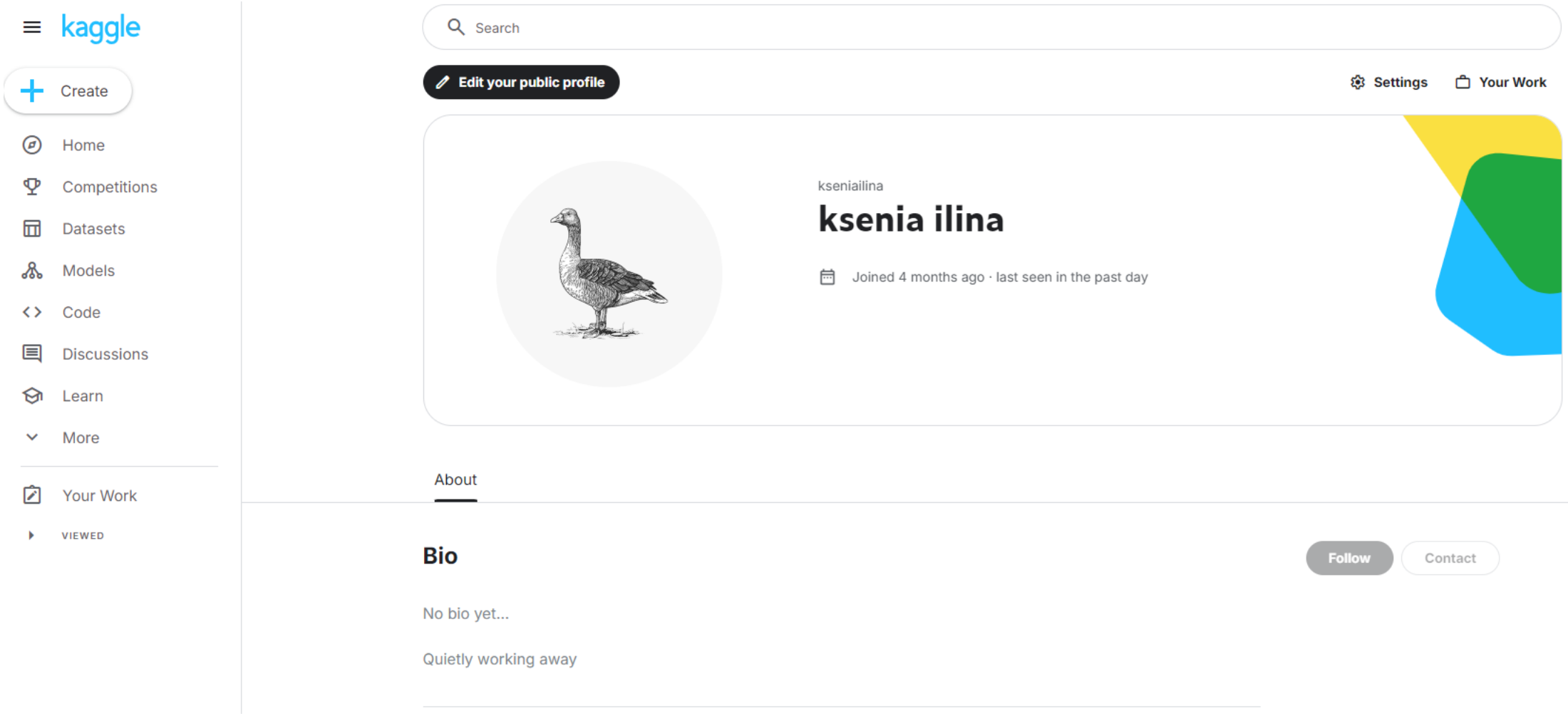
Цели и задачи

- Участие в соревновании на платформе Kaggle, посвященном предсказанию задержек рейсов.

- Проанализировать предоставленные данные, построить модели машинного обучения

- Улучшить их с целью достижения точности предсказаний выше 0.7 (метрика AUC-ROC)

Регистрация на Kaggle



Изучение данных

```
[ ]  data= pd.read_csv('drive/MyDrive/flight_delays_train.csv')
     data.head()
```

	Month	DayofMonth	DayOfWeek	DepTime	UniqueCarrier	Origin	Dest	Distance	dep_delayed_15min
0	c-8	c-21	c-7	1934	AA	ATL	DFW	732	N
1	c-4	c-20	c-3	1548	US	PIT	MCO	834	N
2	c-9	c-2	c-5	1422	XE	RDU	CLE	416	N
3	c-11	c-25	c-6	1015	OO	DEN	MEM	872	N
4	c-10	c-7	c-6	1828	WN	MDW	OMA	423	Y

Вывод первых 5 строк датасета

Размер датасета: (100000, 9)

Данные: месяц, день месяца, день недели, DepTime, UniqueCarrier, Origin, Dest, дистанция, задержка – целевая переменная.

```
data.info()
```

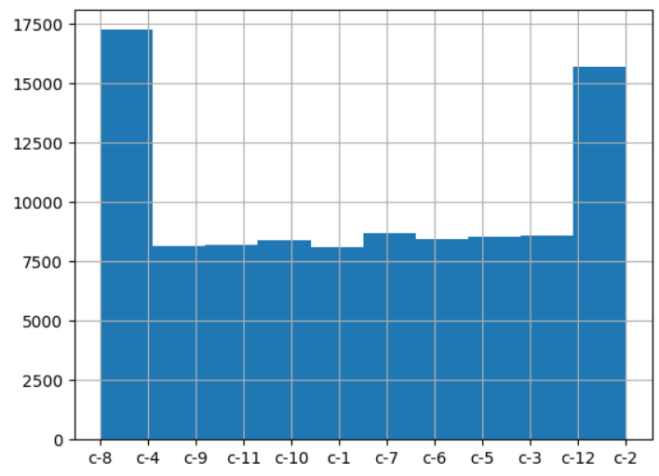
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 9 columns):
 #   Column             Non-Null Count   Dtype 
---  ------             --------------   ----- 
 0   Month              100000 non-null  object
 1   DayofMonth         100000 non-null  object
 2   DayOfWeek          100000 non-null  object
 3   DepTime            100000 non-null  int64 
 4   UniqueCarrier      100000 non-null  object
 5   Origin             100000 non-null  object
 6   Dest               100000 non-null  object
 7   Distance           100000 non-null  int64 
 8   dep_delayed_15min  100000 non-null  object
dtypes: int64(2), object(7)
memory usage: 6.9+ MB
```

Вывод информации о данных каждого столбца

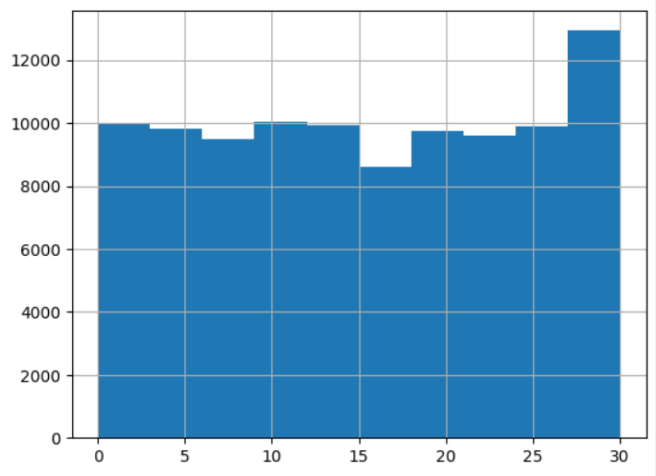
Только 2 столбца содержат целочисленный тип данных. Остальные столбцы необходимо преобразовать.

Изучение данных: гистограммы

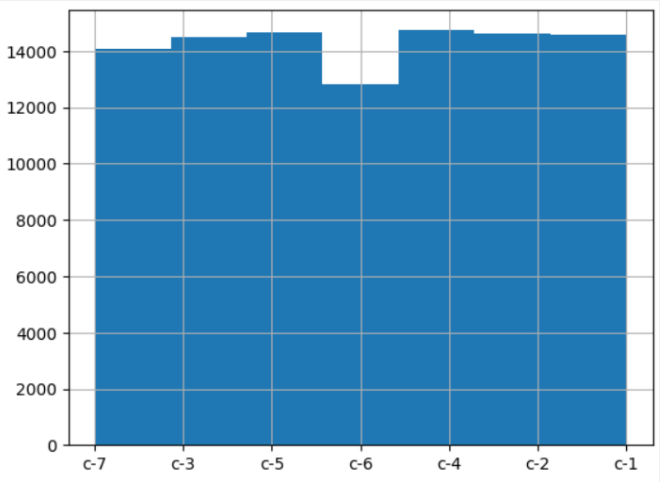
Month



DayofMonth

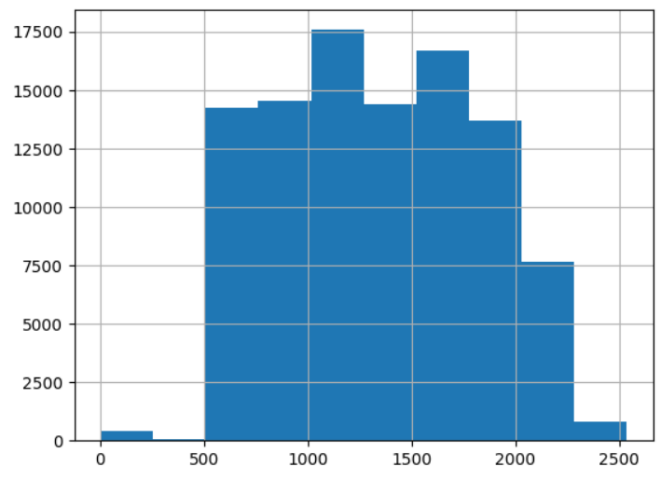


DayOfWeek

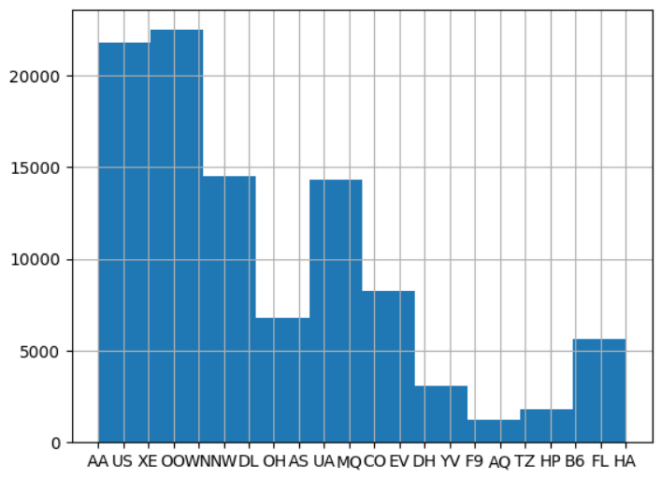


Изучение данных : гистограммы

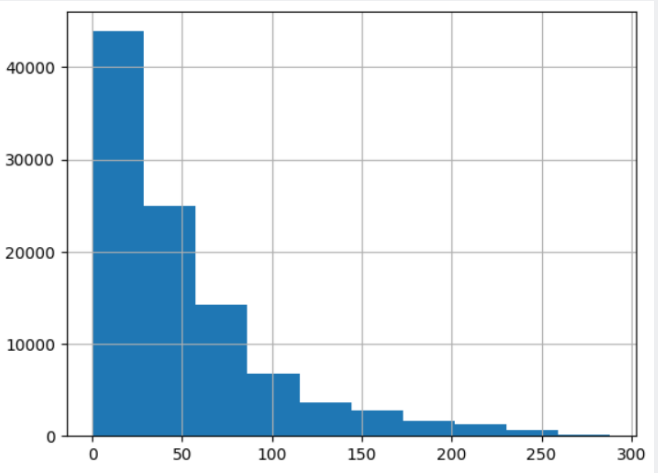
DepTime



UniqueCarrier

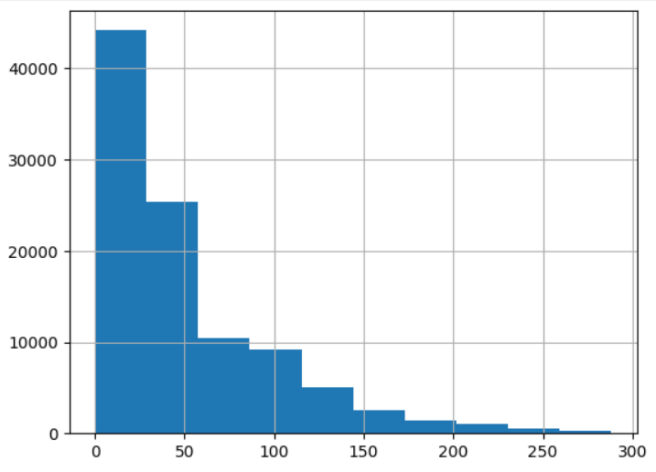


Origin

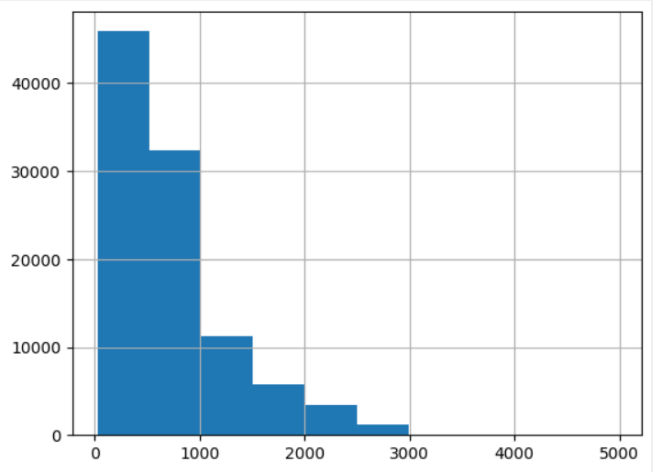


Изучение данных : гистограммы

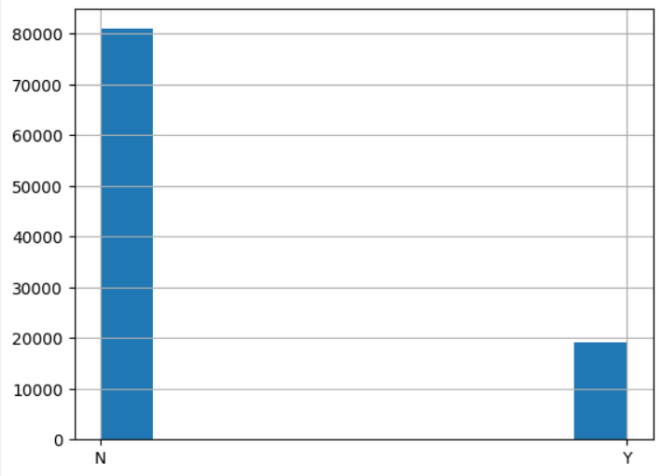
Dest



Distance

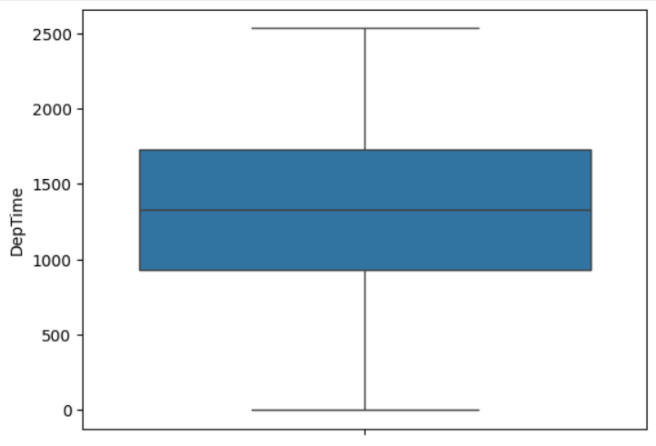


dep_delayed_15min

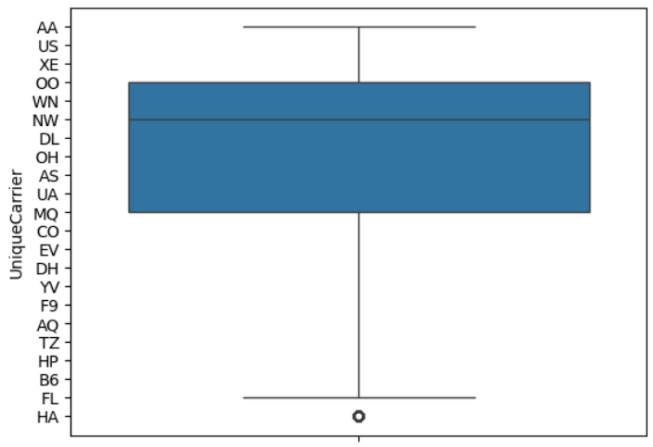


Изучение данных: диаграмма boxplot

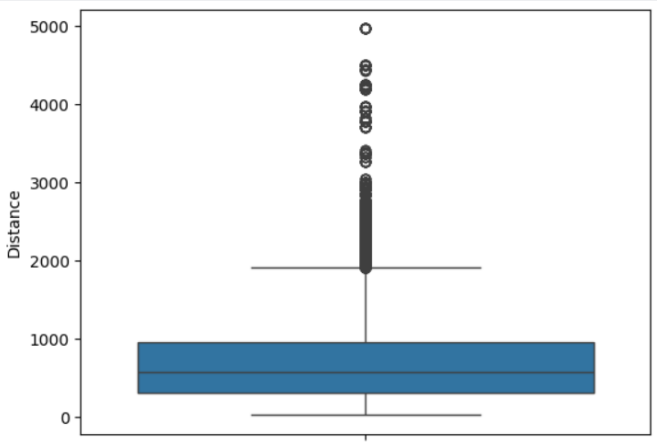
DepTime



UniqueCarrier



Distance



Изучение данных

```
data.isnull().sum()
```

	0
Month	0
DayofMonth	0
DayOfWeek	0
DepTime	0
UniqueCarrier	0
Origin	0
Dest	0
Distance	0
dep_delayed_15min	0

dtype: int64

Вывод информации о пропусках

Столбцы не содержат пропусков, следовательно, проводить этап очистки данных от пропусков не требуется.

Кодирование категориальных признаков

```
[3]  d_Month={'c-1':1,'c-2':2,'c-3':3,'c-4':4,'c-5':5,'c-6':6,'c-7':7,'c-8':8,'c-9':9,'c-10':10,'c-11':11,'c-12':12}
     data['Month']=data['Month'].apply(lambda x:d_Month[x])

     d_DayofMonth={DayofMonth: i for i, DayofMonth in enumerate(data['DayofMonth'].unique())}
     data['DayofMonth']=data['DayofMonth'].map(d_DayofMonth)

     d_DayOfWeek={'c-1':1,'c-2':2,'c-3':3,'c-4':4,'c-5':5,'c-6':6,'c-7':7}
     data['DayOfWeek']=data['DayOfWeek'].apply(lambda x:d_DayOfWeek[x])

     d_UniqueCarrier={UniqueCarrier: i for i, UniqueCarrier in enumerate(data['UniqueCarrier'].unique())}
     data['UniqueCarrier']=data['UniqueCarrier'].map(d_UniqueCarrier)

     d_Origin = {Origin: i for i, Origin in enumerate(data['Origin'].unique())}
     data['Origin'] = data['Origin'].map(d_Origin)

     d_Dest = {Dest: i for i, Dest in enumerate(data['Dest'].unique())}
     data['Dest'] = data['Dest'].map(d_Dest)

     d_dep_delayed_15min={'Y':0,'N':1}
     data['dep_delayed_15min']=data['dep_delayed_15min'].apply(lambda x:d_dep_delayed_15min[x])
```

Категориальные признаки были преобразованы в целые числа.

Предобработка данных

Кодирование категориальных признаков

```
[ ]  data[['Month',  'DayofMonth', 'DayOfWeek',  'DepTime',  'UniqueCarrier',  'Origin',
        'Dest', 'Distance', 'dep_delayed_15min']].corr(method='kendall',numeric_only=True)
```

	Month	DayofMonth	DayOfWeek	DepTime	UniqueCarrier	Origin	Dest	Distance	dep_delayed_15min
Month	1.000000	-0.002555	0.003603	-0.000984	0.007052	-0.002200	-0.001379	0.003969	-0.014379
DayofMonth	-0.002555	1.000000	0.019271	0.002411	-0.002826	-0.001494	0.001412	0.004028	0.002231
DayOfWeek	0.003603	0.019271	1.000000	0.006914	0.006462	-0.001308	-0.000210	0.008162	-0.009456
DepTime	-0.000984	0.002411	0.006914	1.000000	0.005074	-0.059880	0.074335	-0.024369	-0.202473
UniqueCarrier	0.007052	-0.002826	0.006462	0.005074	1.000000	-0.025785	0.008062	-0.016330	-0.015000
Origin	-0.002200	-0.001494	-0.001308	-0.059880	-0.025785	1.000000	-0.117516	-0.093042	0.036790
Dest	-0.001379	0.001412	-0.000210	0.074335	0.008062	-0.117516	1.000000	-0.088604	-0.014844
Distance	0.003969	0.004028	0.008162	-0.024369	-0.016330	-0.093042	-0.088604	1.000000	-0.010030
dep_delayed_15min	-0.014379	0.002231	-0.009456	-0.202473	-0.015000	0.036790	-0.014844	-0.010030	1.000000

Корреляционная матрица показала, что день месяца и день недели слабо влияют на целевую переменную.

Предобработка данных

```
[ ]  scaler = StandardScaler()
     X_scaled = scaler.fit_transform(X)
```

масштабирование данных

```
[ ]  x_columns=['Month', 'DepTime',  'UniqueCarrier',
                 'Origin',  'Dest', 'Distance']
     X=data[x_columns]
     y=data['dep_delayed_15min']
```

```
[ ]  X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=32)
```

1) Берём за X столбцы 'Month', 'DepTime', 'UniqueCarrier', 'Origin', 'Dest', 'Distance' , столбцы 'DayofMonth' и 'DayOfWeek' не берём, так как их влияние на целевую переменную слабое.
2) Берём за y целевой столбец 'dep_delayed_15min'
3) Делим данные на обучающую и тестовую выборки

Построение моделей

Логистическая регрессия

```
[ ] lr = LogisticRegression()
    lr.fit(X_train, y_train)
```

```
LogisticRegression
LogisticRegression()
```

Градиентный бустинг

```
[ ] gb = GradientBoostingClassifier()
    gb.fit(X_train, y_train)
```

```
GradientBoostingClassifier
GradientBoostingClassifier()
```

Случайный лес

```
[ ] rf = RandomForestClassifier()
    rf.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier()
```

XGBoost

```
[ ] xgb = XGBClassifier()
    xgb.fit(X_train, y_train)
```

```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=None,
              enable_categorical=False, eval_metric=None, feature_types=None,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=None,
              max_cat_threshold=None, max_cat_to_onehot=None,
              max_delta_step=None, max_depth=None, max_leaves=None,
              min_child_weight=None, missing=nan, monotone_constraints=None,
              multi_strategy=None, n_estimators=None, n_jobs=None,
              num_parallel_tree=None, random_state=None, ...)
```

Оценка качества

Перекрестная проверка

```
[ ]  print('Cross-validation scores:')
     print(f'Logistic Regression: {lr.score(X_test, y_test)}')
     print(f'Random Forest: {rf.score(X_test, y_test)}')
     print(f'Gradient Boosting: {gb.score(X_test, y_test)}')
     print(f'XGBoost: {xgb.score(X_test, y_test)}')
```

```
Cross-validation scores:
Logistic Regression: 0.80945
Random Forest: 0.81825
Gradient Boosting: 0.8172
XGBoost: 0.82115
```

Метрика AUC-ROC

```
[ ]  lr_auc = roc_auc_score(y_test, lr.predict_proba(X_test)[:, 1])
     rf_auc = roc_auc_score(y_test, rf.predict_proba(X_test)[:, 1])
     gb_auc = roc_auc_score(y_test, gb.predict_proba(X_test)[:, 1])
     xgb_auc = roc_auc_score(y_test, xgb.predict_proba(X_test)[:, 1])

     print('AUC-ROC scores:')
     print(f'Logistic Regression: {lr_auc}')
     print(f'Random Forest: {rf_auc}')
     print(f'Gradient Boosting: {gb_auc}')
     print(f'XGBoost: {xgb_auc}')
```

```
AUC-ROC scores:
Logistic Regression: 0.6921575646157263
Random Forest: 0.7517692136846081
Gradient Boosting: 0.7248366024197563
XGBoost: 0.7380825304495986
```

Внутренний слайд вариант 6

Настройка гиперпараметров модели с использованием метода RandomizedSearchCV

Из 4 моделей лучшие результаты показала модель Random Forest, поэтому RandomizedSearchCV будут применятся для неё.

Метрика AUC-ROC модели Random Forest = 0.7511997863299824

Метрика AUC-ROC для RandomizedSearchCV = 0.755961036834571

Лучшие результаты показала модель RandomizedSearchCV, метрика AUC-ROC выше 0.7, требуемая точность достигнута.

```
AUC-ROC scores:
Logistic Regression: 0.6921575646157263
Random Forest: 0.7511997863299824
Gradient Boosting: 0.724836602419756З
XGBoost: 0.7380825304495986
```

```
[12] # RandomizedSearchCV
param_distributions = {'n_estimators': range(100, 1000, 50), 'max_depth': range(5, 30, 2)}
random_search = RandomizedSearchCV(estimator=rf, param_distributions=param_distributions, cv=5, scoring='roc_auc', n_iter=20)
random_search.fit(X_train, y_train)
```

```
RandomizedSearchCV
best_estimator_: RandomForestClassifier
RandomForestClassifier
```

```
[18] print('RandomizedSearchCV')
print(f'Cross-validation scores: {random_search.score(X_test, y_test)}')
rs_auc = roc_auc_score(y_test,random_search.predict_proba(X_test)[:, 1])
print(f'AUC-ROC scores: {rs_auc}')
```

```
RandomizedSearchCV
Cross-validation scores: 0.755961036834571
AUC-ROC scores: 0.755961036834571
```

Оптимизация модели

Настройка гиперпараметров модели с использованием метода RandomizedSearchCV

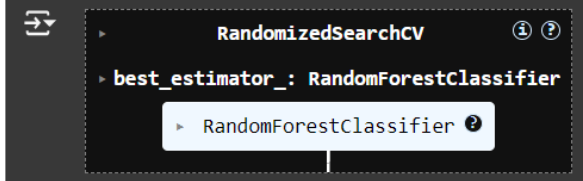
Из 4 моделей лучшие результаты показала модель Random Forest, поэтому RandomizedSearchCV будут применятся для неё.

Метрика AUC-ROC модели Random Forest = 0.7511997863299824

Метрика AUC-ROC для RandomizedSearchCV = 0.755961036834571

Лучшие результаты показала модель RandomizedSearchCV, метрика AUC-ROC выше 0.7, требуемая точность достигнута.

```
[12] # RandomizedSearchCV
     param_distributions = {'n_estimators': range(100, 1000, 50), 'max_depth': range(5, 30, 2)}
     random_search = RandomizedSearchCV(estimator=rf, param_distributions=param_distributions, cv=5, scoring='roc_auc', n_iter=20)
     random_search.fit(X_train, y_train)
```




```
[18] print('RandomizedSearchCV')
     print(f'Cross-validation scores: {random_search.score(X_test, y_test)}')
     rs_auc = roc_auc_score(y_test,random_search.predict_proba(X_test)[:, 1])
     print(f'AUC-ROC scores: {rs_auc}')

RandomizedSearchCV
Cross-validation scores: 0.755961036834571
AUC-ROC scores: 0.755961036834571
```

ИНСТИТУТ ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
УНИВЕРСИТЕТА ИННОПОЛИС

Спасибо за внимание!

Контакты

+79371530799

ksenya.ilina.1@gmail.com

Сайт