Отчет

"Система проката велосипедами"

Поля данных

datetime - ежечасно дата + отметка времени

season - 1 = весна, 2 = лето, 3 = осень, 4 = зимний

holiday - считается праздником или нет

workingday - будни

weather - 1: Ясно, малооблачно, облачно, пасмурно, дождь

2: туман + пасмурно, туман + дождь, туман + дождь, туман

3: легкий снег, Небольшой дождь + Гроза + малооблачно, Небольшой дождь + рассеянные облака

4: Сильный Дождь + Лед + Дождь + Туман, Снег + Туман

temp - температура

atemp - "ощущаемая" температура

humidity - относительная влажность

windspeed - скорость ветра

casual - число незарегистрированных пользователей аренды

registered - количество зарегистрированных аренды

count - количество общая аренда

Training set:

состоит из первых 19 дней каждого месяца

Test set:

состоит из последних 20 дней каждого месяца

Требуемый результат:

**Необходимо предсказать количество велосипедов необходимых для аренды в течении каждого часа**

Преобразования

1 Необходимо изменить представление даты

train['month'] = pd.DatetimeIndex(train.datetime).month

train['day'] = pd.DatetimeIndex(train.datetime).dayofweek

train['hour'] = pd.DatetimeIndex(train.datetime).hour

2 Такие поля как 'casual', 'registered' не влияют зависимость событий, поэтому удалим их

train = train.drop(['datetime','casual','registered'], axis = 1)

3 Выбор алгоритма произведем на основании cross-validation

3.1 Подготовим данные для cross-validation, разделим train set на train\_target и train\_data

train\_target = train['count'].values

train\_data = train.drop(['count'],axis = 1).values

3.2 Сформируем список подходящих алгоритмов

Random Forest Regressor

Extra Trees Regressor

Gradient Boosting Regressor

SVR

Ridge

Random Forest Regressor

Обоснование выбора алгоритма

1 Алгоритм выбирается на основе счета (score) точности обучения

Random Forest Regressor

train score: 0.979, test score: 0.863

train score: 0.980, test score: 0.840

train score: 0.979, test score: 0.860

**Extra Trees Regressor**

**train score: 1.000, test score: 0.864**

**train score: 1.000, test score: 0.854**

**train score: 1.000, test score: 0.860**

Gradient Boosting Regressor

train score: 0.807, test score: 0.788

train score: 0.788, test score: 0.771

train score: 0.809, test score: 0.796

SVR(kernel='rbf',C=10,gamma=.001)

train score: 0.396, test score: 0.391

train score: 0.410, test score: 0.401

train score: 0.413, test score: 0.383

Ridge

train score: 0.342, test score: 0.332

train score: 0.324, test score: 0.350

train score: 0.337, test score: 0.337

2 Выбор параметров запуска алгоритма Extra Trees Regressor

Количество деревьев в лесу.

0.847 (+/-0.007) for {'n\_estimators': 10}

0.862 (+/-0.006) for {'n\_estimators': 100}

0.863 (+/-0.006) for {'n\_estimators': 500}

**нет необходимости применять более 100 в 'n\_estimators'**

**Вывод**

Требуемый результат: Необходимо предугадать количество велосипедов необходимых для аренды в течении каждого часа

Для задачи выявления требуемого количества велосипедов было проанализировано 6 различных алгоритмов обучения на основе тестовых данных.

Extra Trees Regressor показал лучший результат обучения на train score, так и на test score.

Для алгоритма был выбран параметр 'n\_estimators' = 100, потому что большее число деревьев не влияет на качество предсказания результата, но замедляет работу, которое затягивает время работы программы.