Ola Bike Ride Request Forecast using ML

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sb
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from xgboost import XGBRegressor
from sklearn.linear_model import LinearRegression, Lasso, Ridge
from sklearn.ensemble import RandomForestRegressor

import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv('ola.csv')
df.head()
df.shape
df.info()
df.describe().T
parts = df["datetime"].str.split(" ", n=2, expand=True)
df["date"] = parts[0]
df["time"] = parts[1].str[:2].astype('int')
df.head()
parts = df["date"].str.split("-", n=3, expand=True)
df["day"] = parts[0].astype('int')
df["month"] = parts[1].astype('int')
df["year"] = parts[2].astype('int')
df.head()
```

```
from datetime import datetime
import calendar


def weekend_or_weekday(year, month, day):

    d = datetime(year, month, day)
    if d.weekday() > 4:
        return 0
    else:
        return 1


df['weekday'] = df.apply(lambda x:
weekend_or_weekday(x['year'],
x['month'], x['day']), axis=1)
df.head()
def am_or_pm(x):
    if x > 11:
        return 1
    else:
        return 0


df['am_or_pm'] = df['time'].apply(am_or_pm)
df.head()
from datetime import date
import holidays


def is_holiday(x):
```

```
        india_holidays = holidays.country_holidays('IN')

        if india_holidays.get(x):
                return 1
        else:
                return 0


df['holidays'] = df['date'].apply(is_holiday)
df.head()
df.drop(['datetime', 'date'],
                axis=1,
                inplace=True)
df.isnull().sum()
features = ['day', 'time', 'month']

plt.subplots(figsize=(15, 10))
for i, col in enumerate(features):
        plt.subplot(2, 2, i + 1)
        df.groupby(col).mean()['count'].plot()
plt.show()
features = ['season', 'weather', 'holidays',\
                        'am_or_pm', 'year', 'weekday']

plt.subplots(figsize=(20, 10))
for i, col in enumerate(features):
        plt.subplot(2, 3, i + 1)
        df.groupby(col).mean()['count'].plot.bar()
plt.show()
features = ['temp', 'windspeed']
```

```
plt.subplots(figsize=(15, 5))
for i, col in enumerate(features):
plt.subplot(1, 2, i + 1)
sb.distplot(df[col])
plt.show()
features = ['temp', 'windspeed']


plt.subplots(figsize=(15, 5))
for i, col in enumerate(features):
plt.subplot(1, 2, i + 1)
sb.boxplot(df[col])
plt.show()
num_rows = df.shape[0] - df[df['windspeed']<32].shape[0]
print(f'Number of rows that will be lost if we remove outliers is equal to {num_rows}.')
features = ['humidity', 'casual', 'registered', 'count']


plt.subplots(figsize=(15, 10))
for i, col in enumerate(features):
        plt.subplot(2, 2, i + 1)
        sb.boxplot(df[col])
plt.show()
sb.heatmap(df.corr() > 0.8,
                annot=True,
                cbar=False)
plt.show()
df.drop(['registered', 'time'], axis=1, inplace=True)
df = df[(df['windspeed'] < 32) & (df['humidity'] > 0)]
features = df.drop(['count'], axis=1)
target = df['count'].values
```

```
X_train, X_val, Y_train, Y_val = train_test_split(features,
        target,
        test_size = 0.1,
        random_state=22)
X_train.shape, X_val.shape
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_val = scaler.transform(X_val)
from sklearn.metrics import mean_absolute_error as mae
models = [LinearRegression(), XGBRegressor(), Lasso(),
                RandomForestRegressor(), Ridge()]

for i in range(5):
        models[i].fit(X_train, Y_train)

        print(f'{models[i]} : ')

        train_preds = models[i].predict(X_train)
        print('Training Error : ', mae(Y_train, train_preds))

        val_preds = models[i].predict(X_val)
        print('Validation Error : ', mae(Y_val, val_preds))
        print()
```