Scopul principal al proiectului este analiza preturilor uber dintr-o baza de date care contine informatii despre curse in vederea realizarii unui model pentru separarea pretului mediu per km pe anumite intervale orare.

```
import pandas as pd

def date_parser(date_string):
    return pd.datetime.strptime(date_string, '%Y-%m-%d %H:%M:%S.%f')

df = pd.read_excel(r'C:\Users\user\Desktop\uber.xlsx', parse_dates = ['data realizare cursa'], converters = {'data realizare cursa':date_parser})

#eliminam comenzile anulate
filtered_df = df[df["Comanda anulata?"] == "Nu"]



df=filtered_df

#df['price_per_km'] = df['pret lira'] / df['distanta km']

#salvam datele cu o coloana adaugata in acelasi fisier excel

#df.to_excel(r'C:\Users\user\Desktop\uber.xlsx', index=False)



# cream coloana  'day_of_week' care contine ziua din saptamana convertita la numar (Monday=0, Tuesday=1, ..., Sunday=6)
df['day_of_week'] = df['data realizare cursa'].dt.dayofweek

# procedam la fel pentru ora
```

```
df['hour'] = df['data realizare cursa'].dt.hour

# lista pentru intervale orare
hour_interval_dfs = []

# definire intervale
hour_intervals = [(0, 3), (3, 6), (6, 9), (9, 12), (12, 15), (15, 18), (18, 21), (21, 24)]

# bucla peste intervale
for start_hour, end_hour in hour_intervals:
    # creaza nou df
    hour_interval_df = df[(df['hour'] >= start_hour) & (df['hour'] < end_hour)]
    # adauga
    hour_interval_dfs.append(hour_interval_df)

# creem o noua lista
working_days_dfs = []
weekend_dfs = []

# bucla peste df
for hour_df in hour_interval_dfs:
    # df separat pentru zilele lucratoare (Monday=0, Tuesday=1, ..., Friday=4)
    working_days_df = hour_df[~hour_df['day_of_week'].isin([5, 6])]
    # adaugare
    working_days_dfs.append(working_days_df)
    # la fel pt weekend (Saturday=5, Sunday=6)
    weekend_df = hour_df[hour_df['day_of_week'].isin([5, 6])]
    # adauga
```

```
    weekend_dfs.append(weekend_df)

# ca exemplu, se pot accesa si astfel
#print(working_days_dfs[0])  # 00:00-03:00 zi lucratoare
#print(weekend_dfs[1])  # weekend 03:00-06:00

#weekend_dfs.to_excel(r'C:\Users\user\Desktop\uber2.xlsx', index=False)
#working_days_dfs.to_excel(r'C:\Users\user\Desktop\uber3.xlsx', index= False)

writer = pd.ExcelWriter(r'C:\Users\user\Desktop\weekend_dfs.xlsx', engine='xlsxwriter')

# fiecare df in alt sheet
for i, df in enumerate(weekend_dfs):
    df.to_excel(writer, sheet_name=f'Interval {i}')

writer.save()

writer = pd.ExcelWriter(r'C:\Users\user\Desktop\working_days_dfs.xlsx', engine='xlsxwriter')

# la fel
for i, df in enumerate(working_days_dfs):
    df.to_excel(writer, sheet_name=f'Interval {i}')

writer.save()
```

pana aici am realizat salvarea datelor pe intervale in fisierele excel denumite "working_days_dfs" si "weekend_dfs"

```
import pandas as pd
import numpy as np



# citim
weekend_dfs = pd.read_excel(r'C:\Users\user\Desktop\weekend_dfs.xlsx', sheet_name=None)
working_days_dfs = pd.read_excel(r'C:\Users\user\Desktop\working_days_dfs.xlsx',
sheet_name=None)



weekend = [0]*8
working_days = [0]*8

for i in range(8):
    weekend[i] = weekend_dfs['Interval ' + str(i)]
    working_days[i] = working_days_dfs['Interval ' + str(i)]

for i in range(8):
    weekend[i]['year'] = weekend[i]['data realizare cursa'].dt.year
    working_days[i]['year'] = working_days[i]['data realizare cursa'].dt.year

print(weekend[0])
print(working_days[5])

# initializam matrice goala
weekend_means = np.zeros((7,8))
```

```
# bucla peste ani
for i, year in enumerate(range(2009, 2016)):
    # bucla peste intervale
    for j, data in enumerate(weekend):
        # media pentru an si interval
        mean = data[data['year'] == year]['price_per_km'].mean()
        # asignare valoare la pozitie
        weekend_means[i, j] = mean

weekend_means = pd.DataFrame(weekend_means,
                             columns=['Interval 0-3', 'Interval 3-6','Interval 6-9','Interval 9-12','Interval 12-15','Interval 15-18','Interval 18-21', 'Interval 21-24'],
                             index=[2009, 2010, 2011, 2012, 2013, 2014, 2015])


print(weekend_means)


# aceeasi procedura pentru zilele lucratoare
working_days_means = np.zeros((7,8))

for i, year in enumerate(range(2009, 2016)):

    for j, data in enumerate(working_days):
        mean = data[data['year'] == year]['price_per_km'].mean()
        working_days_means[i, j] = mean

working_days_means = pd.DataFrame(working_days_means,
```

columns=['Interval 0-3', 'Interval 3-6','Interval 6-9','Interval 9-12','Interval 12-15','Interval 15-18','Interval 18-21', 'Interval 21-24'],

index=[2009, 2010, 2011, 2012, 2013, 2014, 2015])

writer = pd.ExcelWriter(r'C:\Users\user\Desktop\weekend_means.xlsx', engine='xlsxwriter')

weekend_means.to_excel(writer, sheet_name='Weekend Means')

writer.save()

writer = pd.ExcelWriter(r'C:\Users\user\Desktop\working_days_means.xlsx', engine='xlsxwriter')

working_days_means.to_excel(writer, sheet_name='working_days Means')

writer.save()

urmeaza sa operam pe datele din working_days_means si weekend_means pentru a realiza predictii pentru urmatorii 10 ani (2016-2025)

scriptul aferent:

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import train_test_split

from statsmodels.tsa.arima_model import ARIMA

import pandas as pd

import numpy as np

import openpyxl

weekend_means = pd.read_excel(r'C:\Users\user\Desktop\weekend_means.xlsx')

working_days_means = pd.read_excel(r'C:\Users\user\Desktop\working_days_means.xlsx')

```
#print (weekend_means)
#print(working_days_means)
df=weekend_means

# stabilim variabilele
X = df[["year"]]
y = df.drop("year", axis=1)

#creem model regresie
model = LinearRegression()

# fit model
model.fit(X, y)

# predictii
future_years = range(2016, 2025)
future_years = pd.DataFrame(future_years, columns=["year"])
future_predictions = model.predict(future_years)

#afiseaza
print(future_predictions)

predictions_df=pd.DataFrame(future_predictions)

print(predictions_df)

predictions_df["year"] = future_years
```

```
predictions_df.to_excel(r'C:\Users\user\Desktop\weekend_predictions.xlsx', index=False)



df=working_days_means

# urmam aceeasi procedura si pt zile lucratoare
X = df[["year"]]
y = df.drop("year", axis=1)

model = LinearRegression()

model.fit(X, y)

future_years = range(2016, 2025)
future_years = pd.DataFrame(future_years, columns=["year"])
future_predictions = model.predict(future_years)

print(future_predictions)

predictions_df=pd.DataFrame(future_predictions)

print(predictions_df)

predictions_df["year"] = future_years
predictions_df.to_excel(r'C:\Users\user\Desktop\working_days_predictions.xlsx', index=False)
```

Disponibilitatea informatiei este un aspect vital in secolul 21, avand in vedere evolutia rapida a tehnologiei si a platformelor pe care consumatorii le utilizeaza.

Ca un rezumat privind aplicabilitatea proiectului, putem afirma ca aplicatia uber se poate folosi de modelul de program pentru a realiza predictii pe anumite intervale orare, care sa fie afisate ca informatii generale in interiorul aplicatiei. Astfel, clientii vor avea o anumita siguranta privind un interval de cost pentru o anumita la o anumita ora. Acest lucru poate spori cererea de curse pentru Uber.