```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler
```

```
df = pd.read_csv("/content/dataset.csv (1).zip", parse_dates=[1,2])
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 197428 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   market_id                 196441 non-null  float64
 1   created_at                197428 non-null  datetime64[ns]
 2   actual_delivery_time      197421 non-null  datetime64[ns]
 3   store_id                  197428 non-null  object
 4   store_primary_category    192668 non-null  object
 5   order_protocol            196433 non-null  float64
 6   total_items               197428 non-null  int64
 7   subtotal                  197428 non-null  int64
 8   num_distinct_items        197428 non-null  int64
 9   min_item_price            197428 non-null  int64
 10  max_item_price            197428 non-null  int64
 11  total_onshift_partners    181166 non-null  float64
 12  total_busy_partners       181166 non-null  float64
 13  total_outstanding_orders  181166 non-null  float64
dtypes: datetime64[ns](2), float64(5), int64(5), object(2)
memory usage: 21.1+ MB
```

```
# feature engineering
df['time_taken'] = df['actual_delivery_time'] - df['created_at']
```

```
df['time_taken_mins'] = pd.to_timedelta(df['time_taken']) / pd.Timedelta('60s')
```

```
# some other features we can create as
df['hours'] = df['created_at'].dt.hour
df['day'] = df['created_at'].dt.dayofweek
df.head(3)
```

	market_id	created_at	actual_delivery_time	store_id	store_
0	1.0	2015-02-06 22:24:17	2015-02-06 23:27:16	df263d996281d984952c07998dc54358	
1	2.0	2015-02-10 21:49:25	2015-02-10 22:56:29	f0ade77b43923b38237db569b016ba25	
2	3.0	2015-01-22 20:39:28	2015-01-22 21:09:09	f0ade77b43923b38237db569b016ba25	

```
# drop of some useless features
df.drop(['time_taken','created_at', 'actual_delivery_time', 'store_id'], axis=1, inplace = T
```

```
df.dropna(inplace = True)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 176248 entries, 0 to 197427
Data columns (total 14 columns):
 #   Column                    Non-Null Count   Dtype  
---  ------                    --------------   -----  
 0   market_id                 176248 non-null  float64
 1   store_primary_category    176248 non-null  object 
 2   order_protocol            176248 non-null  float64
 3   total_items               176248 non-null  int64  
 4   subtotal                  176248 non-null  int64  
 5   num_distinct_items        176248 non-null  int64  
 6   min_item_price            176248 non-null  int64  
 7   max_item_price            176248 non-null  int64  
 8   total_onshift_partners    176248 non-null  float64
 9   total_busy_partners       176248 non-null  float64
 10  total_outstanding_orders  176248 non-null  float64
 11  time_taken_mins           176248 non-null  float64
 12  hours                     176248 non-null  int32  
 13  day                       176248 non-null  int32  
dtypes: float64(6), int32(2), int64(5), object(1)
memory usage: 18.8+ MB
```

```
df['store_primary_category'].isnull().sum()
```

```
0
```

```
from sklearn.preprocessing import LabelEncoder

# Create an instance of LabelEncoder
label_encoder = LabelEncoder()

# Fit and transform the column
df['store_primary_category_encoded'] = label_encoder.fit_transform(df['store_primary_categor
```

```
# Check the result
print(df[ 'store_primary_category_encoded'].head())
```

```
0      4
1     46
8     36
14    38
15    38
Name: store_primary_category_encoded, dtype: int64
```

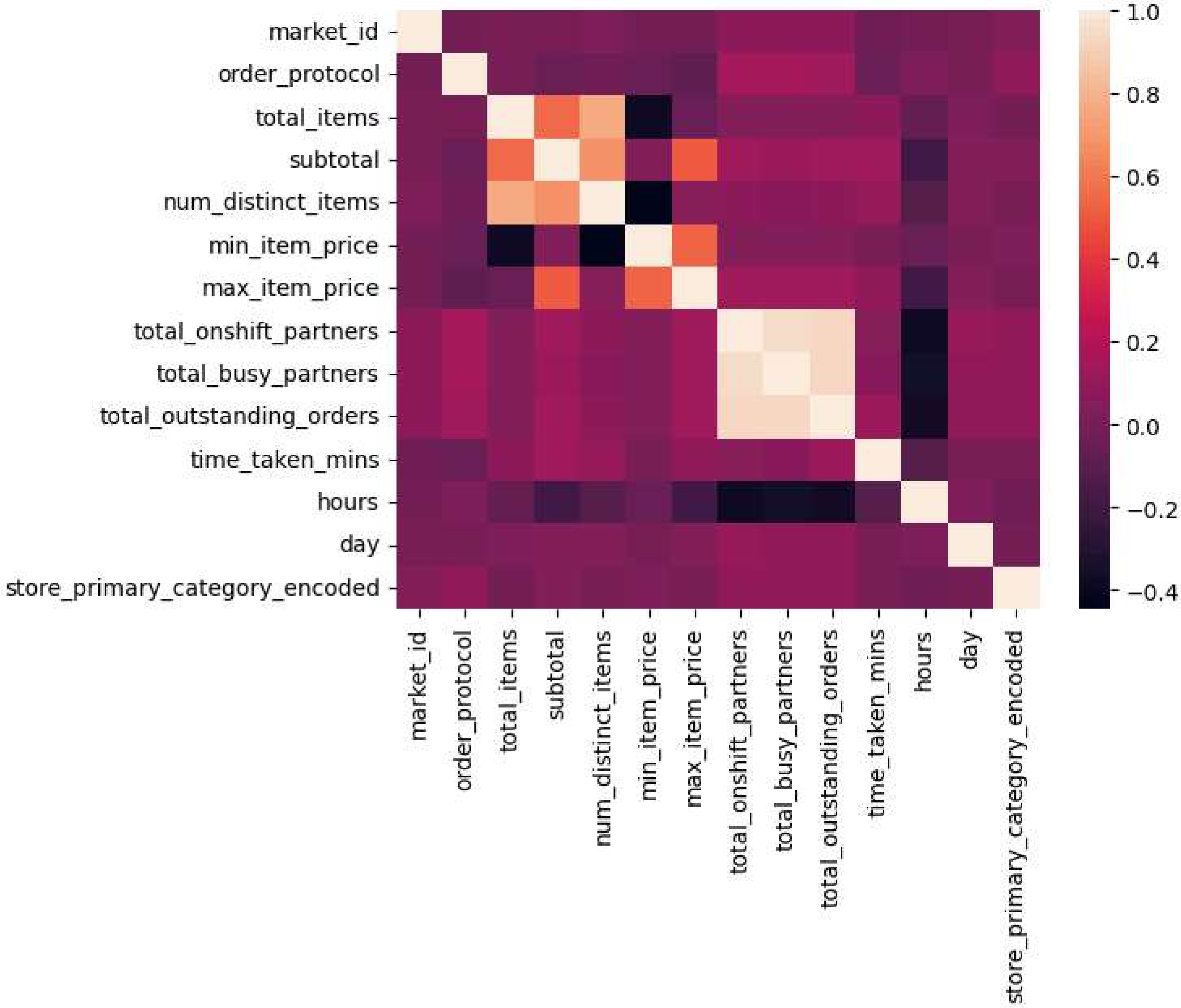
```
df.head()
```

	market_id	store_primary_category	order_protocol	total_items	subtotal	num_distir
0	1.0	american	1.0	4	3441	
1	2.0	mexican	2.0	1	1900	
8	2.0	indian	3.0	4	4771	
14	1.0	italian	1.0	1	1525	
15	1.0	italian	1.0	2	3620	

```
df.drop(['store_primary_category'], axis = 1, inplace = True)
```

```
# check some visualization
sns.heatmap(df.corr())
```

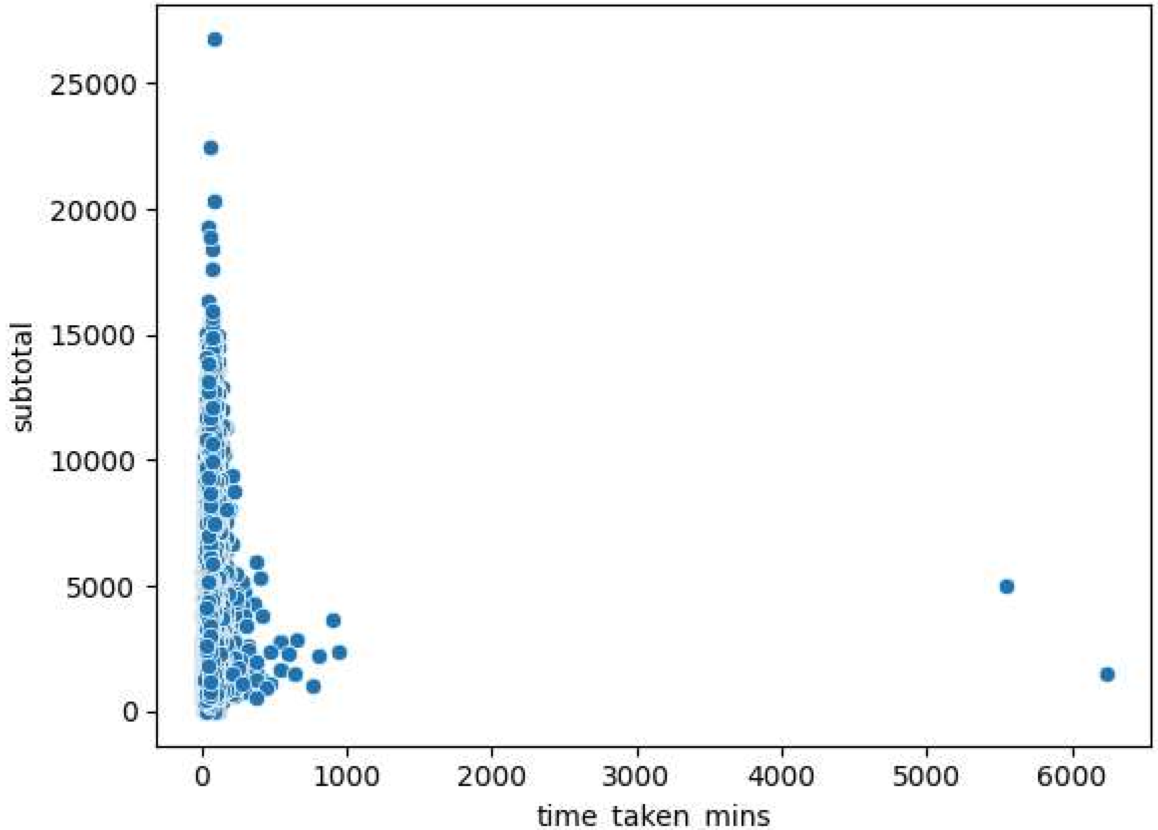
```
<Axes: >
```




```
# Inference : We observe that delivery_time does not show correlation with other feature inc
```

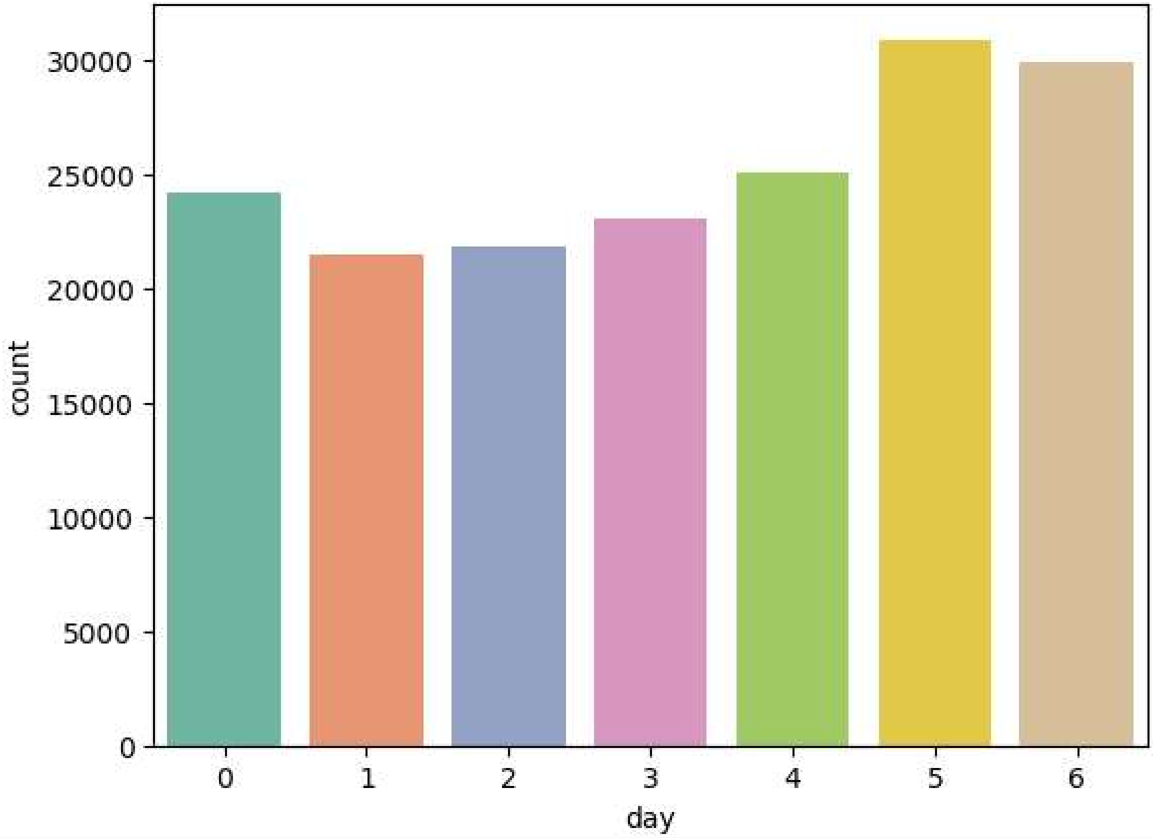
```
sns.scatterplot(x = 'time_taken_mins', y = 'subtotal', data = df)
```

<Axes: xlabel='time_taken_mins', ylabel='subtotal'>



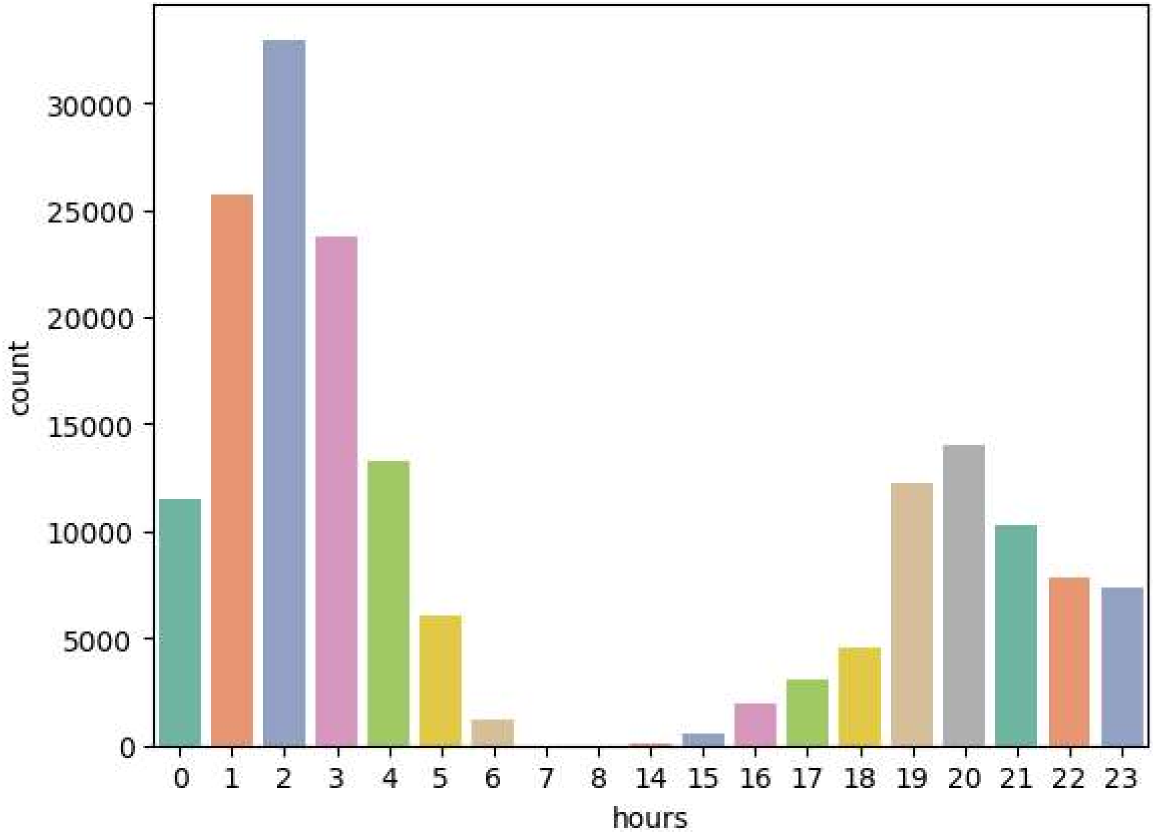

```
sns.countplot(x=df.day, palette='Set2')
plt.show()
```

```
<ipython-input-47-9ece2785a83d>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x=df.day, palette='Set2')
```



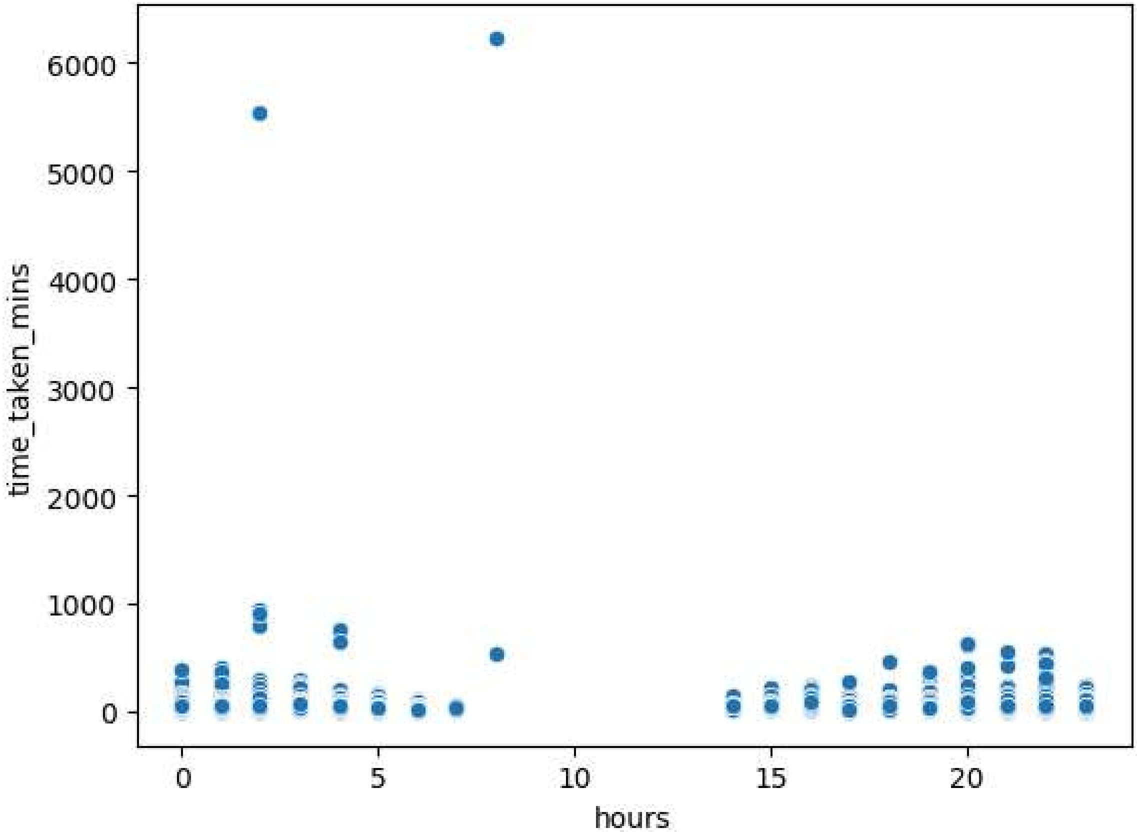

```
sns.countplot(x=df.hours, palette='Set2')
plt.show()
```

```
<ipython-input-48-99d9c83f57f3>:1: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.

  sns.countplot(x=df.hours, palette='Set2')
```




```
sns.scatterplot(x = 'hours', y = 'time_taken_mins', data = df)
```

```
<Axes: xlabel='hours', ylabel='time_taken_mins'>
```




```
df.columns
```

```
Index(['market_id', 'order_protocol', 'total_items', 'subtotal',
       'num_distinct_items', 'min_item_price', 'max_item_price',
       'total_onshift_partners', 'total_busy_partners',
       'total_outstanding_orders', 'time_taken_mins', 'hours', 'day',
       'store_primary_category_encoded'],
      dtype='object')
```

```
# data modeling
y = df['time_taken_mins']
```

```
x = df.drop(['time_taken_mins'], axis = 1)
df.drop(['time_taken_mins'], axis = 1, inplace = True)
```

```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size = 0.2, random_state = 42)
```

```
# random forest
regressor = RandomForestRegressor()
regressor.fit(x_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

```
prediction = regressor.predict(x_test)
mse = mean_squared_error(y_test, prediction)
rmse = mse **.5
print(f'mse : ', mse)
print('rmse : ', rmse)
mae = mean_absolute_error(y_test, prediction)
print('mae :' , mae)
```

```
mse :  319.2601396303887
rmse :  17.86785212694544
mae : 11.67903887073061
```

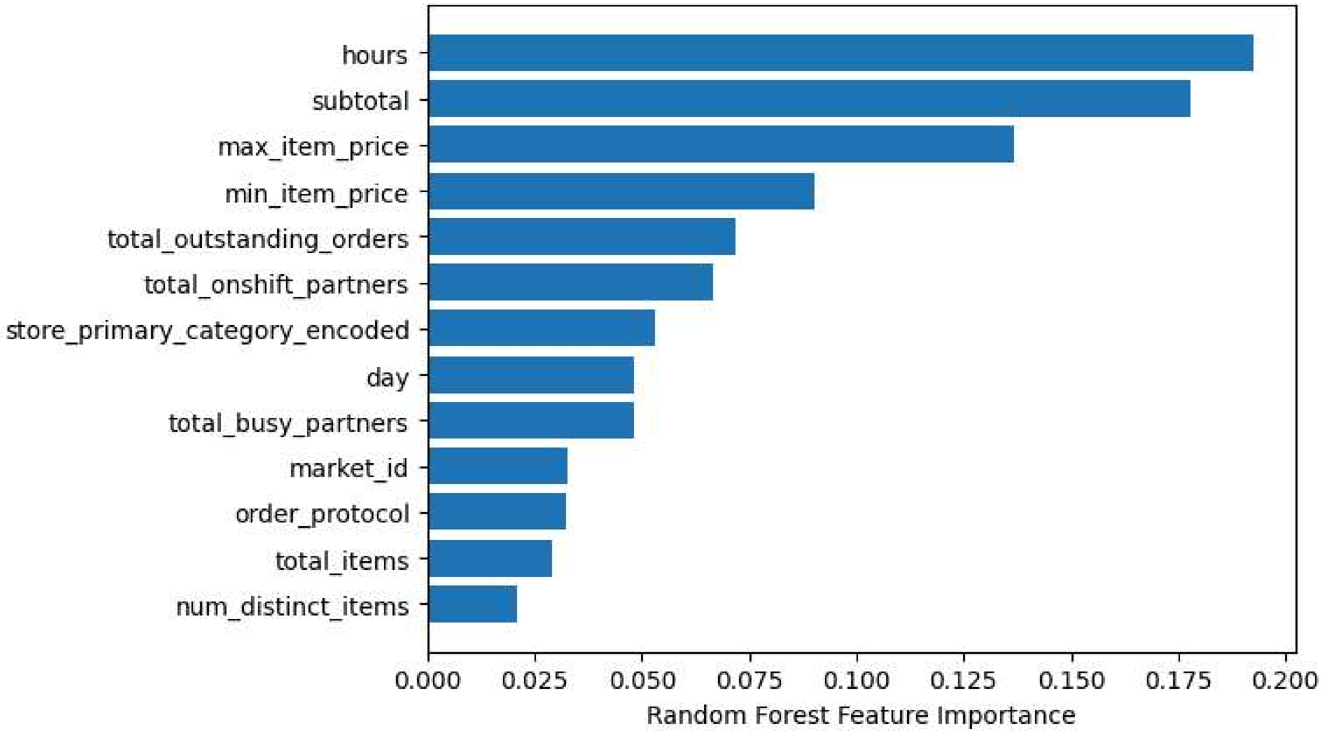
```
r2_score(y_test, prediction)
```

```
0.1414103997165178
```

```
def MAPE(y_actual, y_predicted):
  mape = np.mean(np.abs((y_actual - y_predicted) / y_actual)) * 100
  return mape
print ('mape : ', MAPE(y_test, prediction))
```

```
mape :  26.925932790369334
```

```
# feature importance
sorted_idx = regressor.feature_importances_.argsort()
plt.barh(df.columns[sorted_idx], regressor.feature_importances_[sorted_idx])
plt.xlabel("Random Forest Feature Importance")
plt.show()
```




```
# neural network
from sklearn import preprocessing
scaler = preprocessing.MinMaxScaler()
x_scaled = scaler.fit_transform(x)
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size = 0.2, random_sta


#  when to use minmax or std- max
# minmax : if range is known and there are no outliers
# std-scaler = if range ot known , then the std-scaler use also less prone to outliers


from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
model = Sequential()
model.add(Dense(14, kernel_initializer = 'normal', activation = 'relu'))
model.add(Dense(512, activation = 'relu'))
model.add(Dense(1024, activation = 'relu'))
model.add(Dense(256, activation = 'relu'))
model.add(Dense(1, activation = 'relu'))


from tensorflow.keras.optimizers import Adam
adam = Adam(learning_rate = 0.01)
model.compile(loss = 'mse', optimizer = adam, metrics = ['mse', 'mae'])
history = model.fit(x_train, y_train, epochs = 30, batch_size = 512, verbose = 1, validatior
```

```
Epoch 2/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 20s 67ms/step - loss: 1494.1469 - mae: 15.0795 - mse: 14
Epoch 3/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 27s 97ms/step - loss: 680.1284 - mae: 12.3244 - mse: 680
Epoch 4/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 26s 117ms/step - loss: 1464.8702 - mae: 12.5510 - mse: 14
Epoch 5/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 22s 98ms/step - loss: 1392.8450 - mae: 12.5930 - mse: 13
Epoch 6/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 36s 75ms/step - loss: 782.2251 - mae: 12.2248 - mse: 782
Epoch 7/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 15s 70ms/step - loss: 611.4301 - mae: 12.1938 - mse: 611
Epoch 8/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 20s 67ms/step - loss: 506.5948 - mae: 12.1799 - mse: 506
Epoch 9/30
221/221 ━━━━━━━━━━━━━━━━━━━━ 20s 67ms/step - loss: 711.5001 - mae: 12.2089 - mse: 711
```