```
import plotly.express as px
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from xgboost import plot_importance
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output, State

def plot_features(booster, figsize):
    fig, ax = plt.subplots(1, 1, figsize=figsize)
    return plot_importance(booster=booster, ax=ax)


# Down casts the data entries from int64 to int32 and float64 to float32
# This reduces the size of the records by almost half. (From 134mb to 61mb)
def downcast_dtypes(df):
    float_cols = [c for c in df if df[c].dtype == "float64"]
    int_cols = [c for c in df if df[c].dtype in ["int64", "int32"]]
    df[float_cols] = df[float_cols].astype(np.float32)
    df[int_cols] = df[int_cols].astype(np.int16)
    return df


# Import and clean data (importing csv into pandas)
# Read in .csv files into pandas data frames
train = pd.read_csv('sales_train.csv')
# test = pd.read_csv('test.csv').set_index('ID')
# submission = pd.read_csv('sample_submission.csv')
items = pd.read_csv('items.csv')
# item_cats = pd.read_csv('item_categories.csv')
# shops = pd.read_csv('shops.csv')
items_t = pd.read_csv('items_translated_text.csv')

# Calls the downcasting function
train = downcast_dtypes(train)
# test = downcast_dtypes(test)
# submission = downcast_dtypes(submission)
items = downcast_dtypes(items)


# item_cats = downcast_dtypes(item_cats)
# shops = downcast_dtypes(shops)

# train = train.merge(items, on='item_id')
# replaces the negative price item with the median item_price of all items with the id of 2973 and in shop id 32
# median = train[(train.shop_id == 32) & (train.item_id == 2973) & (train.date_block_num == 4) & (
#           train.item_price > 0)].item_price.median()
```

```
def create_clean_df(train):
    train = train.merge(items, on='item_id')
    train = train.drop(columns='item_name')
    train['date'] = pd.to_datetime(train['date'], format='%d.%m.%Y')

    # Removes outliers from train
    train = train[train.item_price < 90000]
    train = train[train.item_cnt_day < 999]

    train_cnt = train['item_cnt_day']
    train.drop(labels=['item_cnt_day'], axis=1, inplace=True)
    train.insert(6, 'item_cnt_day', train_cnt)

    train_grouped_month = pd.DataFrame(
        train.groupby(['date_block_num', 'shop_id', 'item_category_id', 'item_id', 'item_price'])[
            'item_cnt_day'].sum().reset_index())
    train_grouped_month.rename(columns={'item_cnt_day': 'item_cnt_month'}, inplace=True)
    return train_grouped_month


def create_one_shop_one_item_df(itemid, shopid, train_grouped_month):
    one_shop_df = train_grouped_month[train_grouped_month['shop_id'] == shopid]
    one_shop_one_item_df = one_shop_df[one_shop_df['item_id'] == itemid]
    return one_shop_one_item_df


def create_one_shop_df(shopid, train_grouped_month):
    return train_grouped_month[train_grouped_month['shop_id'] == shopid]


def create_3d_scatter_fig(df):
    return px.scatter_3d(df, x='date_block_num', y='item_price', z='item_cnt_month', color='item_price')


def get_translated_name(itemid):
    return items_t[items_t['item_id'] == itemid]['english_name']


def get_valid_item_list(train_grouped_month, shopid):
    one_shop = train_grouped_month[train_grouped_month['shop_id'] == shopid]
    return list(one_shop.item_id.unique())


def convert_list_to_options_dict(valid_items):
    list_of_dicts = []

    for item in valid_items:
        temp_dict = {'label': item, 'value': item}
```

```
        list_of_dicts.append(temp_dict)
    return list_of_dicts


# cleans data
train_grouped_month = create_clean_df(train)

sample_shop_id = 55
sample_item_id = 492

# creates a data frame containing just one shop with a particular shop_id
one_shop_df = create_one_shop_df(sample_shop_id, train_grouped_month)

# creates a sample figure
fig = create_3d_scatter_fig(one_shop_df)

# gets the list of items that are sold in a particular shop to put into the drop down menu
valid_items = get_valid_item_list(train_grouped_month, sample_shop_id)
valid_dict_list_items = convert_list_to_options_dict(valid_items)

# gets a list of the valid shop_id's to display in the drop down menu
valid_shops = train_grouped_month.shop_id.unique()
valid_dict_list_shops = convert_list_to_options_dict(valid_shops)

# basic stylesheet
external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']
app = dash.Dash(__name__, external_stylesheets=external_stylesheets)
styles = {
    'pre': {
        'border': 'thin lightgrey solid',
        'overflowX': 'scroll'
    }
}

# App layout
app.layout = html.Div([

    html.H1("Sales Forecasting", style={'text-align': 'center'}),

    # contains the graph
    html.Div([
        dcc.Graph(
            id='3d-scatter',
            figure={})]),
    html.Br(),

    # contains the shop id drop down menu
    html.Div(['Shop ID: ',
              dcc.Dropdown(
```

```
            id='shop-dropdown',
            options=valid_dict_list_shops,
            placeholder="Select a Shop ID (0-60)"
        )
        ]),

    # contains the item id drop down menu
    html.Div(['Item ID: ',
        dcc.Dropdown(
            id='item-dropdown',
            options=valid_dict_list_items,
            placeholder="Select an Item ID"
        )
        ]),

    # displays the item name
    html.Div([
        'Item Name: ',
        html.Div(
            id='item_name')]),

    # contains the submit button
    html.Div(html.Button(id='submit-button-state', n_clicks=0, children='Show Sales Graph')),

    html.Br(),
])


# Connects the selected shop_id to the item_id drop down with valid item_id's
@app.callback(
    Output(component_id='item-dropdown', component_property='options'),
    Input('shop-dropdown', 'value')
)
def update_dropdown_option(shop_id_from_dropdown):
    temp_list = get_valid_item_list(train_grouped_month, shop_id_from_dropdown)
    return convert_list_to_options_dict(temp_list)


# Connect the Plotly graphs with Dash drop down Components
@app.callback(
    [Output(component_id='3d-scatter', component_property='figure'),
     Output(component_id='item_name', component_property='children')],
    [Input('submit-button-state', 'n_clicks')],
    [State("shop-dropdown", "value"),
     State("item-dropdown", "value")]
)
def update_graph(n_clicks, input_shop_id, input_item_id):
    update_df = create_one_shop_one_item_df(input_item_id, input_shop_id, train_grouped_month)
    # update_df = create_one_shop_df(input_shop_id, train_grouped_month)
```

```
    return create_3d_scatter_fig(update_df), 'Item Name: '.join(get_translated_name(input_item_id))


# runs the whole thing
if __name__ == '__main__':
    app.run_server(debug=True)
```