Assignment

Name :- V.Sai manideep

Vtu :- 21443

Course Name :- Data visualization

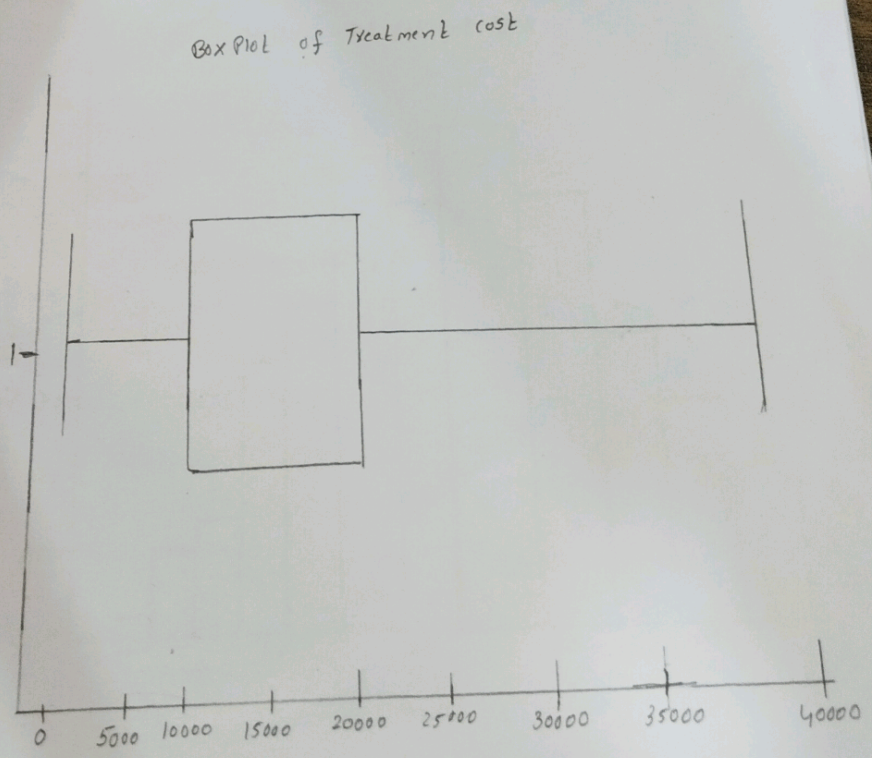
Course Code :- 10212CS214

Slot :- S2L5

1) Explain data abstraction and identify data types (categorical: Gender, continuous: age, cost).

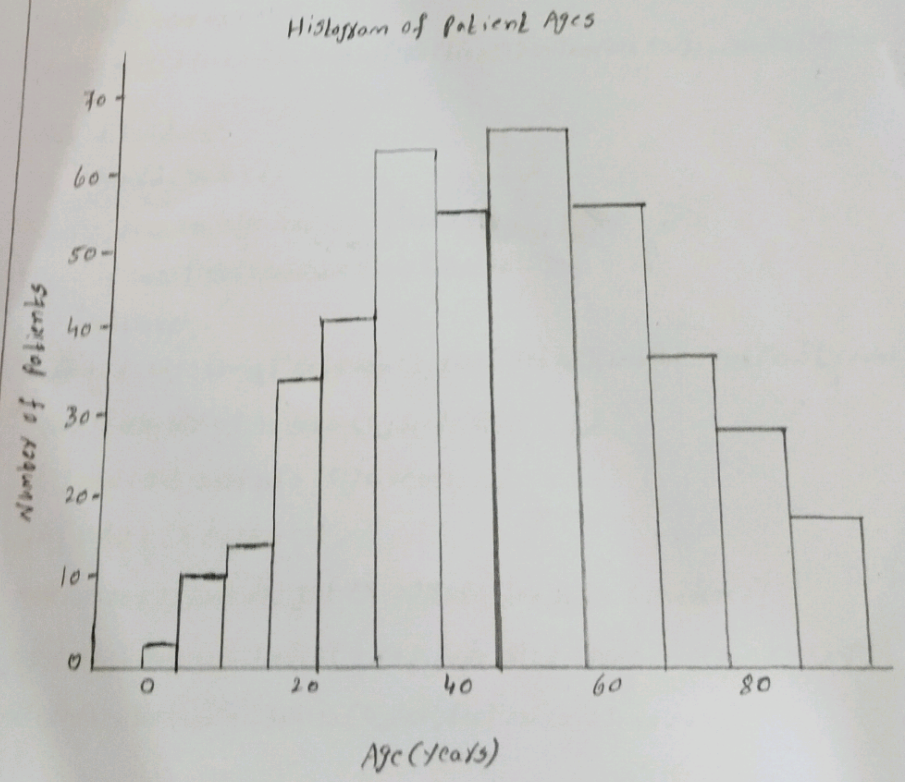
A)

```
fig, ax = plt.subplots(figsize=(8,3))
ax.boxplot(df['Treatmentcost'].dropna(), vert=False)
ax.set_title("Box Plot of Treatment cost")
ax.set_xlabel("Treatment cost")
path = os.path.join(out_dir, "boxplot_treatment_cost.png")
fig.tight_layout()
fig.savefig(path, dpi=150)
plt.close(fig)
saved.append(path)
```



2) Apply EDA Histogram for age, boxplot for treatment cost)

A)
```
fig, ax = plt.subplots(figsize=(8,5))
ax.hist(df['age'].dropna(), bins=15)
ax.set_title("Histogram of patient ages")
ax.set_xlabel("Age (years)")
ax.set_ylabel("Number of patients")
path = os.path.join(out_dir, "hist_age.png")
fig.tight_layout()
fig.savefig(path, dpi=150)
plt.close(fig)
saved.append(path)
```



3) use Network visualization for doctor-patient referrals and text visualization for diagnosis notes.

A) def referral_network_plot(df, out_path = "outputs/network/referral_network.png"):

"""

Build and draw a directed referral network aggregated by counts.

- Nodes: doctors (RefFrom/RefTo)

- Edge weight: number of patients referred from A -> B

"""

os.makedirs(os.path.dirname(out_path), exist_ok=True)

edge_counts = df.groupby(['RefFrom', 'RefTo']).size().reset_index(name='count')

G = nx.DiGraph()

docs = sorted(set(edge_counts['RefFrom']).union(set(edge_counts['RefTo'])))

for d in docs:

G.add_node(d)

for _, row in edge_counts.iterrows():

if row['RefFrom'] == row['RefTo']:

continue

G.add_edge(row['RefFrom'], row['RefTo'], weight=int(row['count']))

pos = nx.spring_layout(G, seed=42, k=0.6)

fig, ax = plt.subplots(figsize=(9, 6))

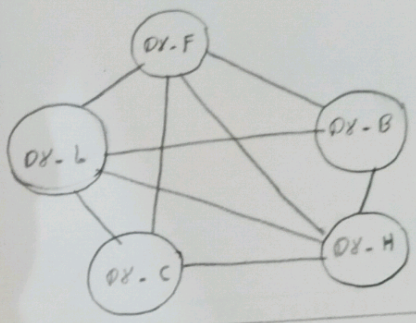
deg = dict(G.degree())

node_sizes = [300 + deg.get(n, 0) * 150 for n in G.nodes()]

nx.draw_network_nodes(G, pos, node_size=node_sizes, ax=ax)]

nx.draw_network_labels(G, pos, font_size=9, ax=ax)

Doctor-patient network

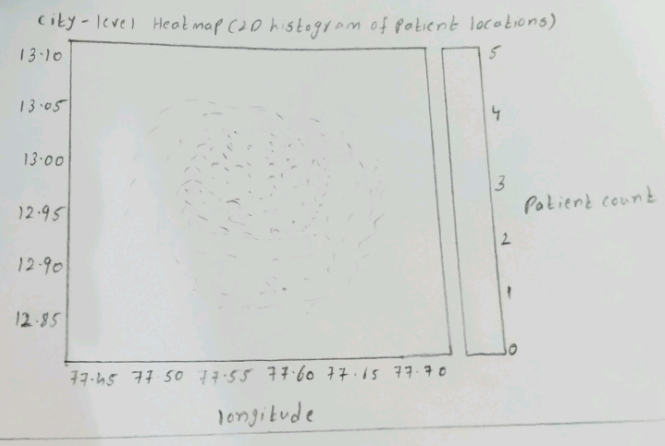


4) map patient addresses on a city-level heat map

A)

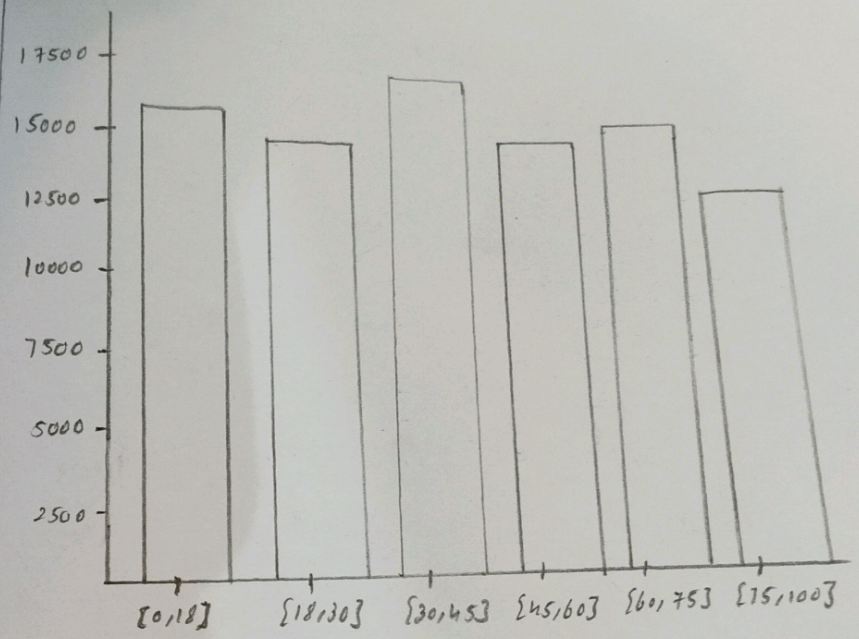
```
def city_heatmap_static(df, out_path="outputs/maps/heatmap_2dhist.
png", bins=50):
os.makedirs(os.path.dirname(out_path), exist_ok=True)
fig, ax = plt.subplots(figsize=(8,6))
x = df['longitude'].dropna()
y = df['latitude'].dropna()
ax.hist2d(x, y, bins=bins)
cbar = plt.colorbar(ax=ax.collections[0], ax=ax)
cbar.set_label('Patient count')
ax.set_title("city-level Heat map (2D histogram)")
ax.set_xlabel("Longitude")
ax.set_ylabel("Latitude")
fig.tight_layout()
fig.savefig(out_path, dpi=150)
plt.close(fig)
```

return out_path



5) Create a hospital performance dashboard?

A)
```
def hospital_performance_dashboard(df, out_dir="outputs/dashboard"):
    os.makedirs(out_dir, exist_ok=True)
    total_patients = int(len(df))
    avg_cost = float(df['Treatment cost'].mean())
    median_cost = float(df['Treatment cost'].median())
    readmission_proxy = float((df['RefFrom'] == df['RefTo']).mean())
    num_doctors = int(df['RefTo'].nunique())
    kpis = {
        'total_patients': total_patients,
        'avg_cost': avg_cost,
        'median_cost': median_cost,
        'readmission_proxy': readmission_proxy,
        'num_doctors': num_doctors
    }
```


17500
15000
12500
10000
7500
5000
2500
[0,18]
[18,30]
[30,45]
[45,60]
[60,75]
[75,100]