```
import pandas as pd
import numpy as np
import plotly.express as px
from plotly.offline import iplot

from sklearn.model_selection import train_test_split, cross_val_score, KFold
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

import warnings
warnings.filterwarnings("ignore")
pd.set_option('future.no_silent_downcasting', True)
pd.options.mode.copy_on_write = "warn"


def add_line(x0 = 0, y0 = 0, x1 = 0, y1 = 0,
             line_color = "#00DFA2", font_color = "#3C486B",
             xposition = "right", text = "Text"):
    fig.add_shape(type='line',
                  x0 = x0,
                  y0 = y0,
                  x1 = x1,
                  y1 = y1 + 2,
                  line = {
                      "color" : line_color,
                      "width" : 3,
                      "dash" : "dashdot"
                  },
                  label={
                      "text" : f"\t{text}: {x1: 0.1f}\t".expandtabs(5),
                      "textposition": "end",
                      "yanchor" :"top",
                      "xanchor" :xposition,
                      "textangle" :0,
                      "font": {
                          "size": 14,
                          "color" :font_color,
                          "family" : "arial"

                      },
                  }
                 )


def custome_layout(title_size = 28, hover_font_size = 16, showlegend = False):
    fig.update_layout(
    showlegend = showlegend,
    title = {
        "font" :{
            "size" :title_size,
            "family" : "tahoma"
        }
    },
    hoverlabel = {
```

```
        "bgcolor" :"#111",
        "font_size" : hover_font_size,
        "font_family" :"arial"
    }

)
```

```
df = pd.read_csv("/content/Salary Data.csv")
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 375 entries, 0 to 374
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   Age                  373 non-null    float64
 1   Gender               373 non-null    object
 2   Education Level      373 non-null    object
 3   Job Title            373 non-null    object
 4   Years of Experience  373 non-null    float64
 5   Salary               373 non-null    float64
dtypes: float64(3), object(3)
memory usage: 17.7+ KB
```

```
df.sample(10, random_state=15)
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
64	26.0	Male	Bachelor's	Junior Accountant	1.0	35000.0
367	41.0	Male	Bachelor's	Senior Product Manager	14.0	150000.0
116	40.0	Female	Bachelor's	Office Manager	15.0	65000.0
176	42.0	Female	PhD	Senior Marketing Manager	18.0	140000.0
239	39.0	Male	Bachelor's	Senior Marketing Specialist	10.0	120000.0
351	31.0	Male	Bachelor's	Junior Marketing Coordinator	3.0	55000.0
253	28.0	Male	Bachelor's	Junior Business Development Associate	2.0	40000.0
254	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
277	34.0	Female	Bachelor's	Junior Financial Analyst	5.0	70000.0
222	33.0	Male	Bachelor's	Senior Product Development Manager	7.0	100000.0

```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	373.0	37.431635	7.069073	23.0	31.0	36.0	44.0	53.0
Years of Experience	373.0	10.030831	6.557007	0.0	4.0	9.0	15.0	25.0
Salary	373.0	100577.345845	48240.013482	350.0	55000.0	95000.0	140000.0	250000.0

```
df.isna().sum()
```

	0
Age	2
Gender	2
Education Level	2
Job Title	2
Years of Experience	2
Salary	2

dtype: int64

```
df[df["Age"].isna()]
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
172	NaN	NaN	NaN	NaN	NaN	NaN
260	NaN	NaN	NaN	NaN	NaN	NaN

```
df.dropna(inplace=True)
df.isna().sum()
```

	0
Age	0
Gender	0
Education Level	0
Job Title	0
Years of Experience	0
Salary	0

dtype: int64

```
df.duplicated().sum()
```

```
np.int64(49)
```

```
df[df.duplicated()].head(15)
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
195	28.0	Male	Bachelor's	Junior Business Analyst	2.0	40000.0
250	30.0	Female	Bachelor's	Junior Marketing Coordinator	2.0	40000.0
251	38.0	Male	Master's	Senior IT Consultant	9.0	110000.0
252	45.0	Female	PhD	Senior Product Designer	15.0	150000.0
253	28.0	Male	Bachelor's	Junior Business Development Associate	2.0	40000.0
254	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
255	44.0	Male	Bachelor's	Senior Software Engineer	14.0	130000.0
256	34.0	Female	Master's	Senior Financial Advisor	6.0	100000.0
257	35.0	Male	Bachelor's	Senior Project Coordinator	9.0	95000.0
258	50.0	Female	PhD	Director of Operations	22.0	180000.0
262	46.0	Male	PhD	Senior Data Scientist	18.0	160000.0
281	41.0	Female	Bachelor's	Senior Project Coordinator	11.0	95000.0
287	35.0	Female	Bachelor's	Senior Marketing Analyst	8.0	85000.0
303	45.0	Male	PhD	Senior Data Engineer	16.0	150000.0
306	49.0	Female	Master's	Director of Marketing	21.0	180000.0

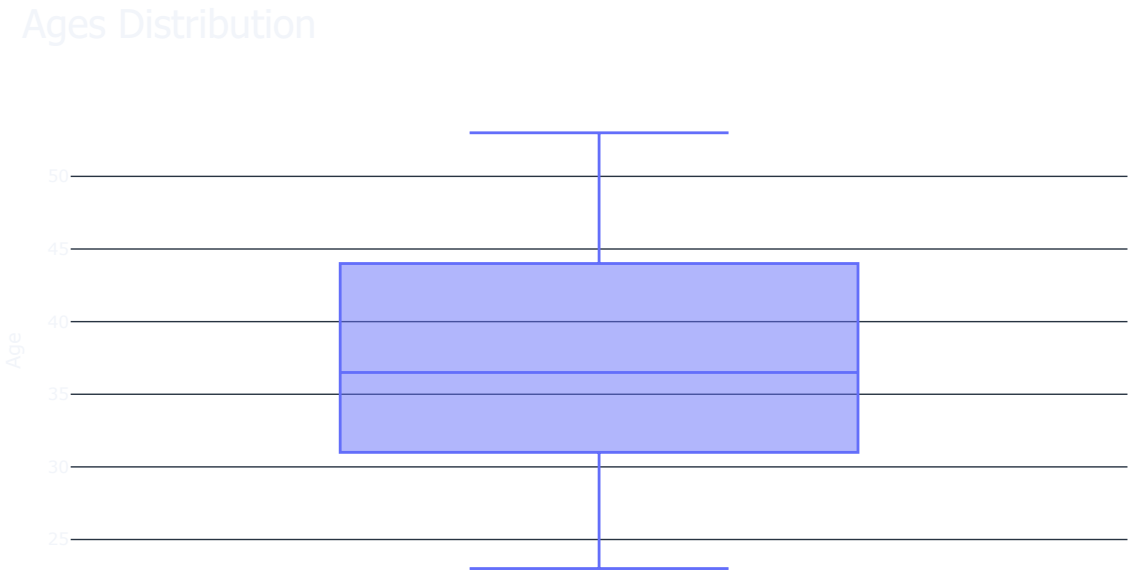
```
df.drop_duplicates(inplace=True)
df.reset_index(inplace=True, drop=True)
```

```
df.head()
```

	Age	Gender	Education Level	Job Title	Years of Experience	Salary
0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0
1	28.0	Female	Master's	Data Analyst	3.0	65000.0
2	45.0	Male	PhD	Senior Manager	15.0	150000.0
3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0
4	52.0	Male	Master's	Director	20.0	200000.0

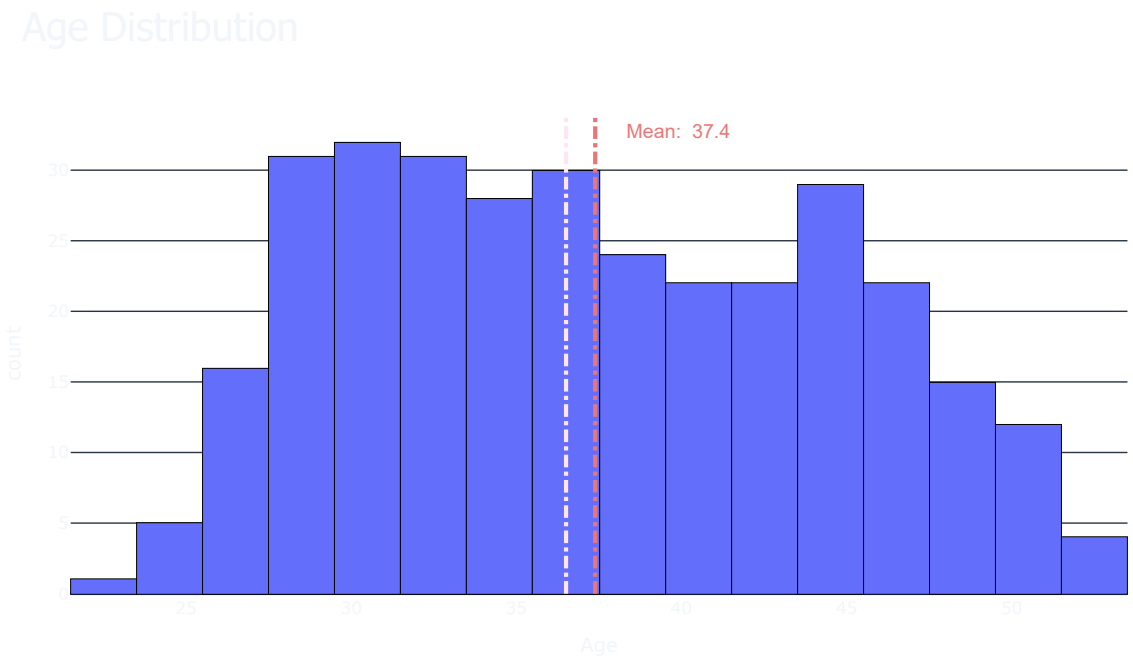
```
mean_of_age = df["Age"].mean()
median_of_age=df["Age"].median()
```

```
fig = px.box(
    y=df["Age"],
    title= "Ages Distribution",
    template="plotly_dark",
    labels={"y" :"Age"},
)
custome_layout()

iplot(fig)
```




```
fig = px.histogram(
    df["Age"],
    nbins=25,
    title="Age Distribution",
    template="plotly_dark",
    labels={"value" : "Age"}
)
custome_layout()
fig.update_traces(
    textfont={
        "size":20,
        "family":"tahoma",
        "color":"#fff"
    },
    hovertemplate="Age : %{x}<br>Frequency:%{y}",
     marker=dict(line=dict(color='#000', width=0.1))
```

```
)


# Adding Mean Line
add_line(x0=mean_of_age, y0=0, x1=mean_of_age, y1=30+2, line_color="#E97777",font_color="#E97777",
         text="Mean", xposition="left")

# Adding Median Line
add_line(x0=median_of_age, y0=0, x1=median_of_age, y1=30+2, line_color="#FFE5F1",
         font_color="#fff", xposition="right", text="Median")

iplot(fig)
```

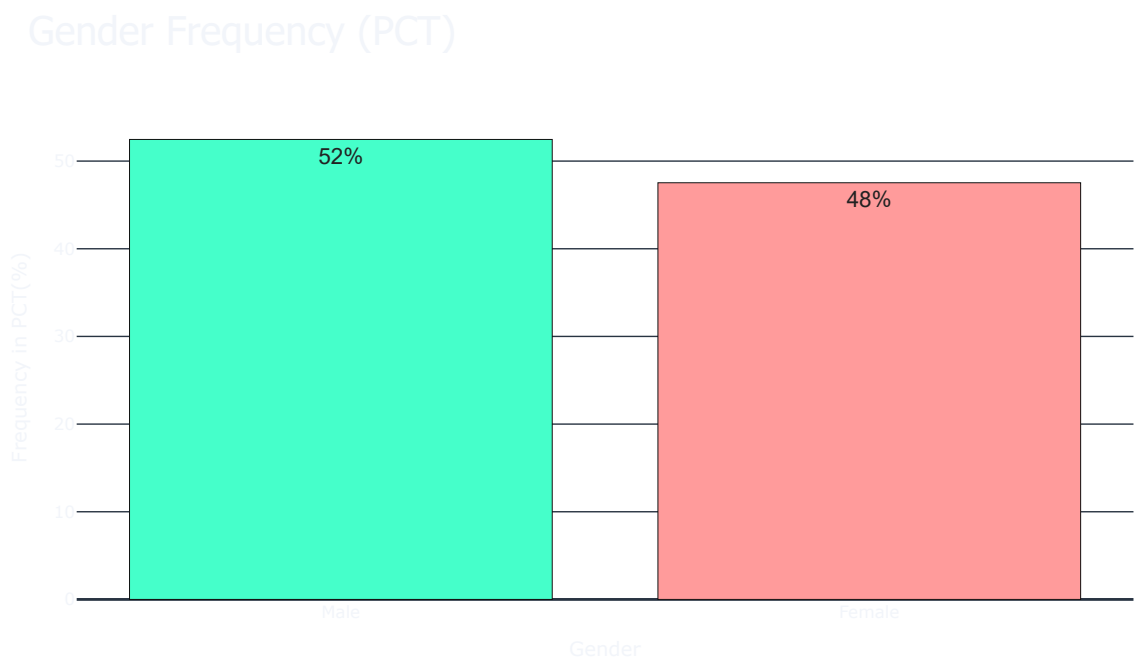



```
gender=df["Gender"].value_counts(normalize=1)*100
gender.apply(lambda x:f"{x:0.2f}%")
```

Gender	proportion
Male	52.47%
Female	47.53%

dtype: object

```
fig = px.bar(data_frame = gender,
             x=gender.index,
             y=gender,
             color = gender.index,
             title="Gender Frequency (PCT)",
             color_discrete_sequence=["#45FFCA","#FF9B9B"],
             labels={"index":"Gender","y":"Frequency in PCT(%)"},
             template="plotly_dark",
             text=gender.apply(lambda x: f"{x:0.0f}%"))
custome_layout()
fig.update_traces(
    textfont={
        "size":16,
        "family":"arial",
        "color":"#222"
    },
    hovertemplate="Gender : %{x}<br>Percentage: %{y:0.1f}%",
)
iplot(fig)
```

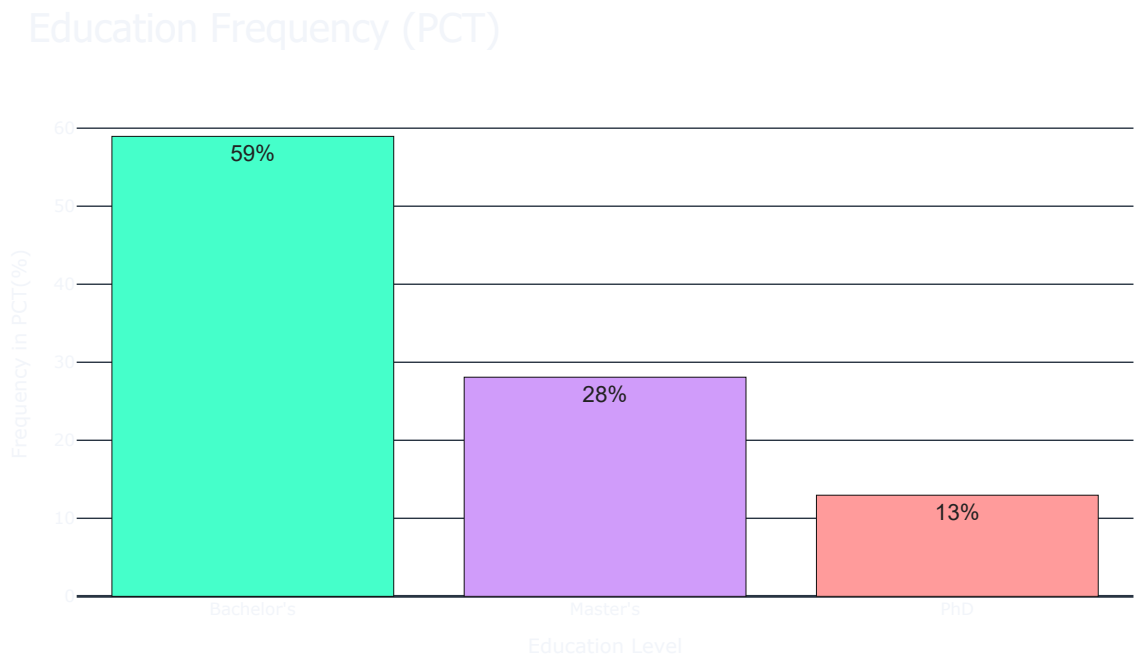



```
education=df["Education Level"].value_counts(normalize=1)*100
education.apply(lambda x: f"{x:0.2f}")
```

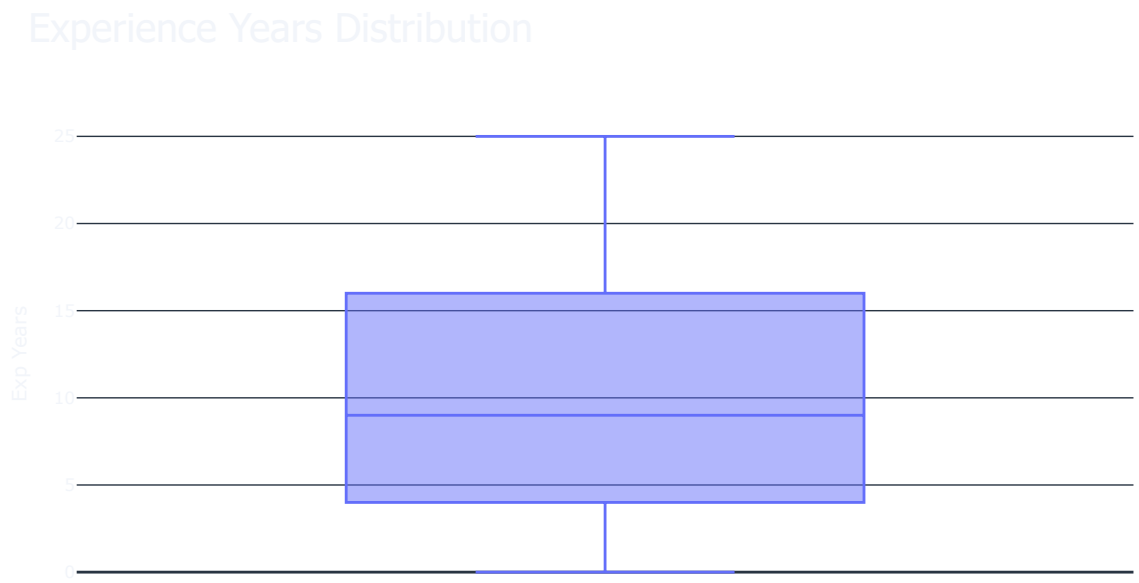
Education Level	proportion
Bachelor's	58.95
Master's	28.09
PhD	12.96

dtype: object

```
fig = px.bar(data_frame = education,
             x = education.index,
             y = education,
             color = education.index,
             title = "Education Frequency (PCT)",
             color_discrete_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],
             labels= {"index" :"Education", "y": "Frequency in PCT(%)"},
             template="plotly_dark",
             text = education.apply(lambda x: f"{x:0.0f}%"))

custome_layout()


fig.update_traces(
    textfont = {
        "size" : 16,
        "family" :"arial",
        "color": "#222"
    },
    hovertemplate = "Education: %{x}<br>Percentage: %{y:0.1f}%",
)

iplot(fig)
```




```
fig = px.box(
    y=df["Years of Experience"],
    title= "Experience Years Distribution",
    template = "plotly_dark",
    labels = {"y":"Exp Years"},
)
custome_layout()
iplot(fig)
```



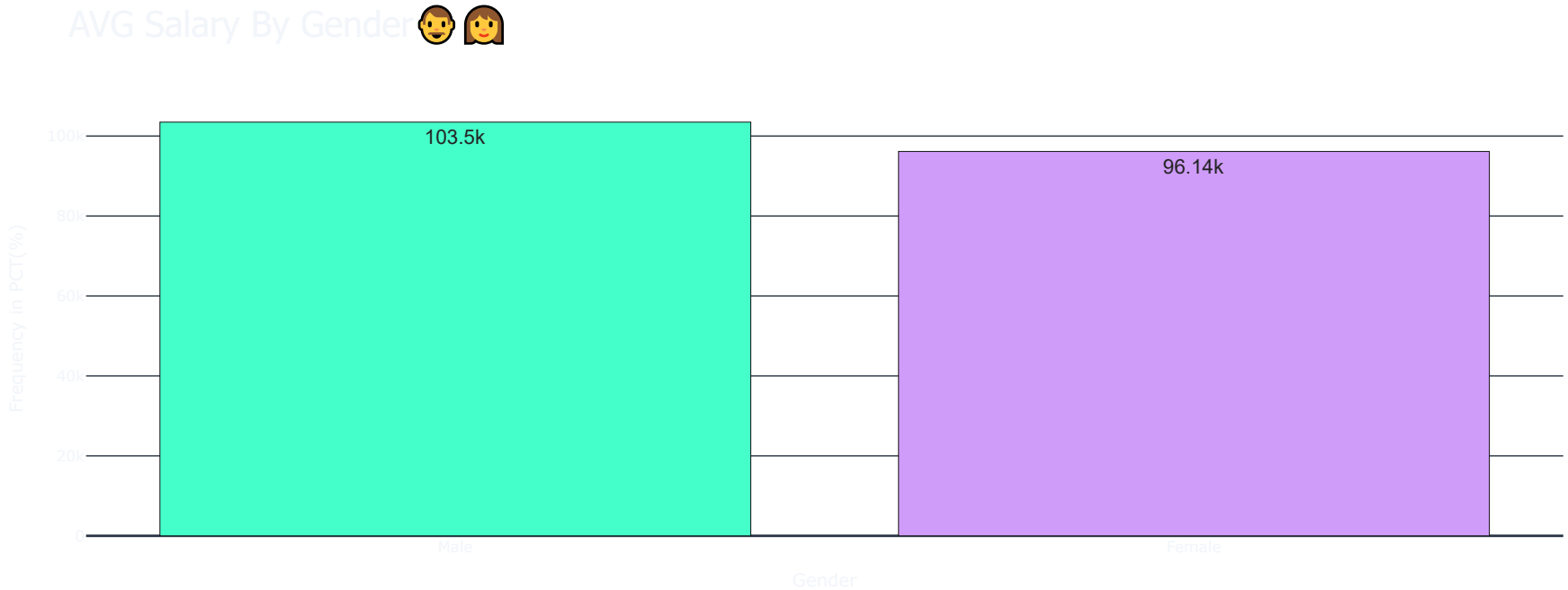

```
salary_by_gender = df.groupby("Gender")["Salary"].mean().sort_values(ascending=False)
salary_by_gender.apply(lambda x: f"${x:,.2f}")
```

	Salary
Gender	
Male	$103,472.65
Female	$96,136.36

dtype: object

```
fig = px.bar(data_frame = salary_by_gender,
             x = salary_by_gender.index,
             y = salary_by_gender,
             color = salary_by_gender.index,
             title = "AVG Salary By Gender👨👩",
             color_discrete_sequence=["#45FFCA", "#D09CFA", "#FF9B9B"],
             labels= {"index" :"Education", "y": "Frequency in PCT(%)"},
             template="plotly_dark",
             text_auto = "0.4s"
            )

custome_layout()
```

```
fig.update_traces(
    textfont = {
        "size" : 16,
        "family" :"arial",
        "color": "#222"
    },
    hovertemplate = "Gender: %{x}<br>Average Salary: $%{y:0.4s}",
)

iplot(fig)
```



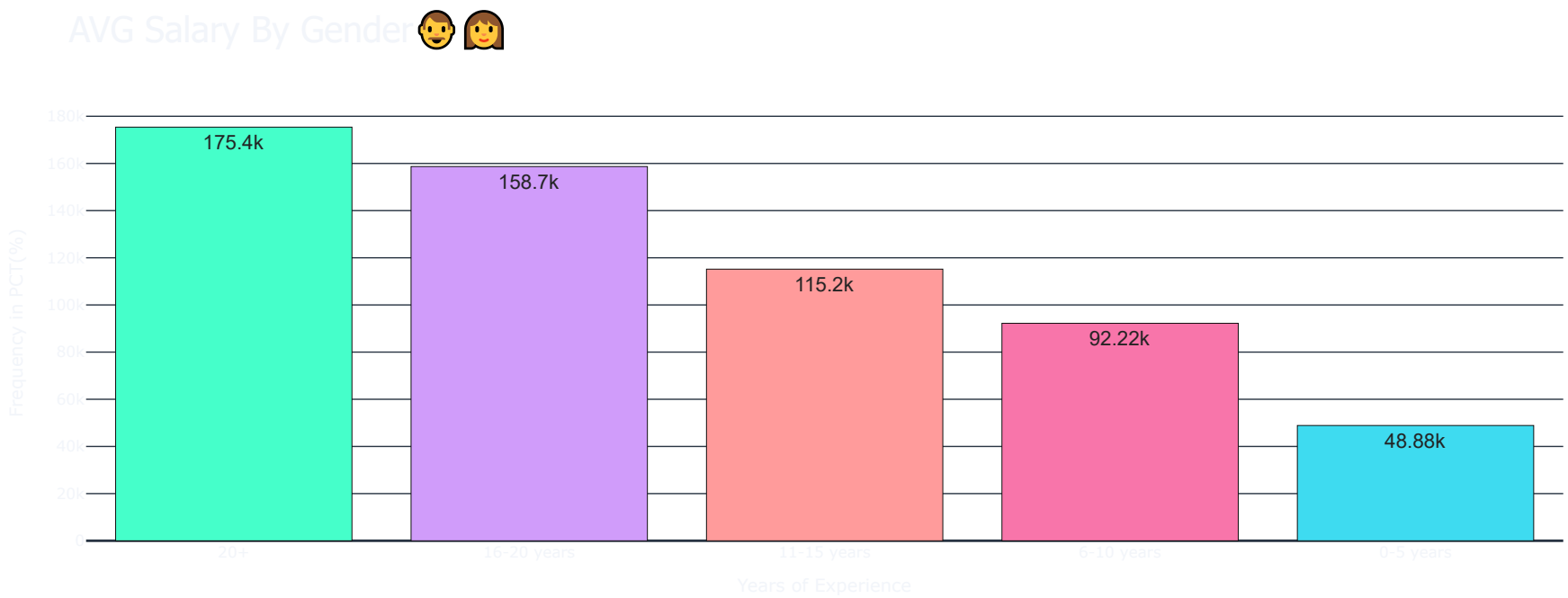

```
def groupping_exp(exp):
  if exp >= 0 and exp <= 5:
    return "0-5 years"
  elif exp > 5 and exp <= 10:
    return "6-10 years"
  elif exp > 10 and exp <= 15:
    return "11-15 years"
  elif exp > 15 and exp <= 20:
    return "16-20 years"
  else:
    return "20+"
```

```
salary_by_exp = df.groupby(df["Years of Experience"].apply(groupping_exp))["Salary"].mean().sort_values(ascending=False)
salary_by_exp.apply(lambda x: f"${x:,.2f}")
```

Years of Experience	Salary
20+	$175,400.00
16-20 years	$158,684.21
11-15 years	$115,178.57
6-10 years	$92,215.19
0-5 years	$48,881.78

dtype: object

```
fig = px.bar(data_frame = salary_by_exp,
             x = salary_by_exp.index,
             y = salary_by_exp,
             color = salary_by_exp.index,
             title = "AVG Salary By Gender👨👩",
             color_discrete_sequence=["#45FFCA", "#D09CFA", "#FF9B9B", "#F875AA", "#3EDBF0"],
             labels= {"index" :"Education", "y": "Frequency in PCT(%)"},
             template="plotly_dark",
             text_auto = "0.4s"
            )

custome_layout()


fig.update_traces(
    textfont = {
        "size" : 16,
        "family" :"arial",
        "color": "#222"
    },
    hovertemplate = "Gender: %{x}<br>Average Salary: $%{y:0.4s}",
)

iplot(fig)
```




```
df_encoded = pd.get_dummies(df, columns=["Education Level"], drop_first=True) *1
df_encoded.head()
```

	Age	Gender	Job Title	Years of Experience	Salary	Education Level_Master's	Education Level_PhD
0	32.0	Male	Software Engineer	5.0	90000.0	0	0
1	28.0	Female	Data Analyst	3.0	65000.0	1	0
2	45.0	Male	Senior Manager	15.0	150000.0	0	1
3	36.0	Female	Sales Associate	7.0	60000.0	0	0
4	52.0	Male	Director	20.0	200000.0	1	0

```
X = df_encoded.drop(columns=["Job Title", "Salary", "Gender"])
y = df_encoded["Salary"]
```

```
X.head()
```

	Age	Years of Experience	Education Level_Master's	Education Level_PhD
0	32.0	5.0	0	0
1	28.0	3.0	1	0
2	45.0	15.0	0	1
3	36.0	7.0	0	0
4	52.0	20.0	1	0

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=90)
```

```
kf = KFold(n_splits=10, shuffle=True, random_state=30)
```

```
rf = RandomForestRegressor(n_estimators=500, random_state=11)
```

```
scores = cross_val_score(rf, X, y, cv=kf)
print(f"Cross Validation Score: {np.mean(scores)*100:0.2f}%")
```

```
Cross Validation Score: 85.81%
```

```
rf.fit(X_train,y_train)
```

```
RandomForestRegressor
RandomForestRegressor(n_estimators=500, random_state=11)
```

```
score = rf.score(X_train, y_train)*100
print(f"Model Score: {np.round(score, 2)}%")
```

```
Model Score: 94.15%
```

```
predicted_salary=np.round(rf.predict(X_test))
```

```
d={
    "Actually_Salary" : y_test,
    "Predicted_Salary" : predicted_salary,
    "error": predicted_salary-y_test
}
predected_df=pd.DataFrame(d)
predected_df.head()
```

	Actually_Salary	Predicted_Salary	error
224	160000.0	152630.0	-7370.0
279	140000.0	130743.0	-9257.0
130	160000.0	178505.0	18505.0
186	100000.0	93598.0	-6402.0
149	175000.0	170956.0	-4044.0

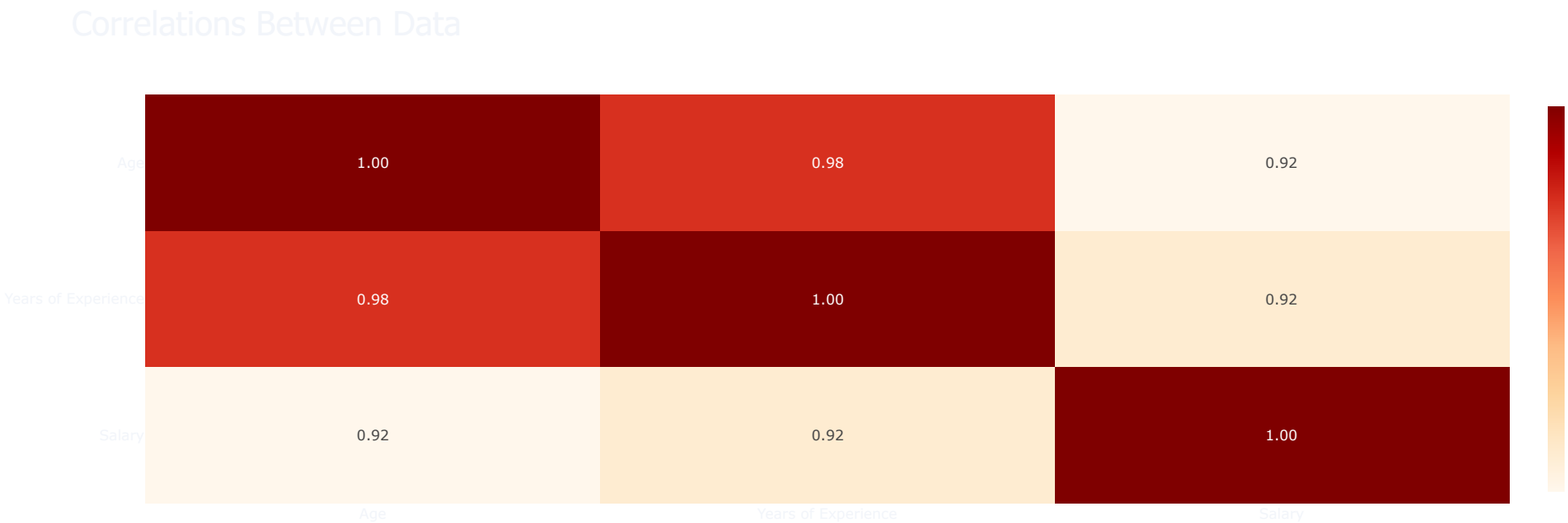
```
score = r2_score(y_test, predicted_salary)*100
print(f"Model Score: {np.round(score, 2)}%")
```

```
Model Score: 93.58%
```

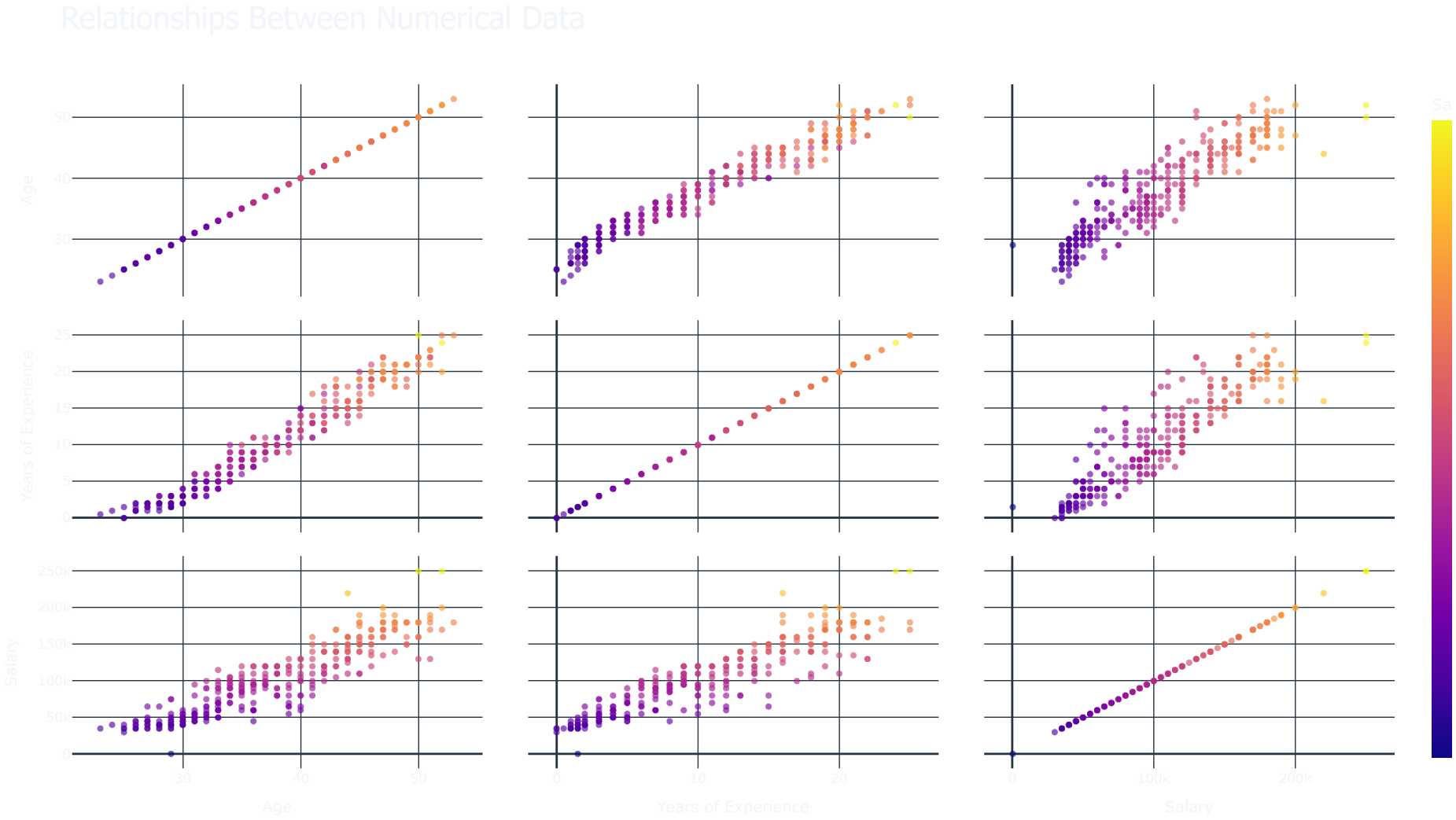
```
rmse = np.sqrt(mean_squared_error(y_test, predicted_salary))
print(f"Error Ratio: {rmse:.3f}")
```

```
Error Ratio: 12844.269
```

```
correlation = df.corr(numeric_only=True)

fig = px.imshow(
    correlation,
    template = "plotly_dark",
    text_auto = "0.2f",
    aspect=1,
    color_continuous_scale="orrd",
    title= "Correlations Between Data"
)

fig.update_layout(
    title = {
        "font" :{
            "size" : 28,
            "family" : "tahoma"
        }
    }
)
iplot(fig)
```




```
fig = px.scatter_matrix(
    df,
    dimensions=df.select_dtypes(include="number").columns,
    height=800,
    color="Salary",
    opacity=0.65,
    title= "Relationships Between Numerical Data",
    template="plotly_dark"

)

fig.update_layout(
    title = {
        "font" :{
            "size" : 28,
            "family" : "tahoma"
        }
    }
)
iplot(fig)
```


Relationships Between Numerical Data
Age
Years of Experience
Salary
30
40
50
0
10
20
0
100k
200k

```
fig = px.box(
    x = df["Education Level"], y = df["Salary"],
    title= "Salary Vs. Education Level",
    template="plotly_dark",
    labels={"x": "Education Level", "y" :"Salary"}
)
```