`VizPool`: A Python Library with Low-Code support for Basic to Advance Static & Interactive Visualizations

Overview

Welcome to getting started with VizPool; in this introduction to VizPooL, you learn how to use vizpool static API for creating static but complex visuals in one line of code. In case you are looking for interactive visuals, please visit VizPool Interactive API

Introduction

In this notebook, you will learn how to perform EDA on the **Tips dataset**. You will create quite complex visuals for the EDA process by simply calling a class method on an instance of the `EDA` class.

In the second section, we will see how to plot machine learning model evaluation metrics by simply calling the class methods on the instance of the `Evaluation` class.

> This notebook is divided into 3 major sections which are as follows:

1. Installations & Imports

```
!pip install --upgrade --quiet vizpool
```

```
#Restart the kernel after package upgrade
#import IPython
#IPython.Application.instance().kernel.do_shutdown(True)
```

```
import vizpool as vizpool
vizpool.__version__
```

```
'0.0.9'
```

```
from src.vizpool.static import EDA
from src.vizpool.static import Evaluation

import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression, SGDClassifier,  LinearRegression
from sklearn.svm import SVC
from sklearn.tree import ExtraTreeClassifier
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder
```

2. Exploratory Data Analysis (EDA)

```
#load the dataset and create a pandas dataframe for analysis
df = sns.load_dataset("tips")
df
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
...	...	...	...	...	...	...	...
239	29.03	5.92	Male	No	Sat	Dinner	3
240	27.18	2.00	Female	Yes	Sat	Dinner	2
241	22.67	2.00	Male	Yes	Sat	Dinner	2
242	17.82	1.75	Male	No	Sat	Dinner	2
243	18.78	3.00	Female	No	Thur	Dinner	2

244 rows × 7 columns

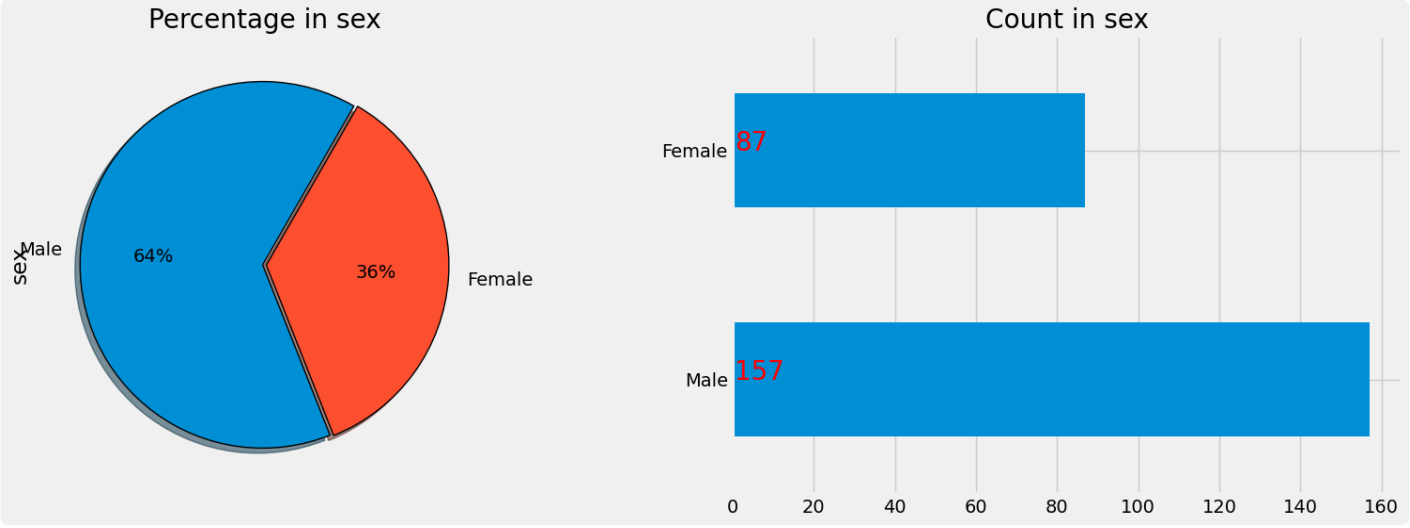
```
# Instantiate the EDA class
tips_eda = EDA(df)
```

> **Note** : Every class method in **VizPooL** returns an `object` which can be stored in a variable to save or use the visual in any other application. refer to the following example;

a. Pie Bar chart

```
plt = tips_eda.pie_bar(hue='sex');
plt.savefig("pie_bar.png")
```

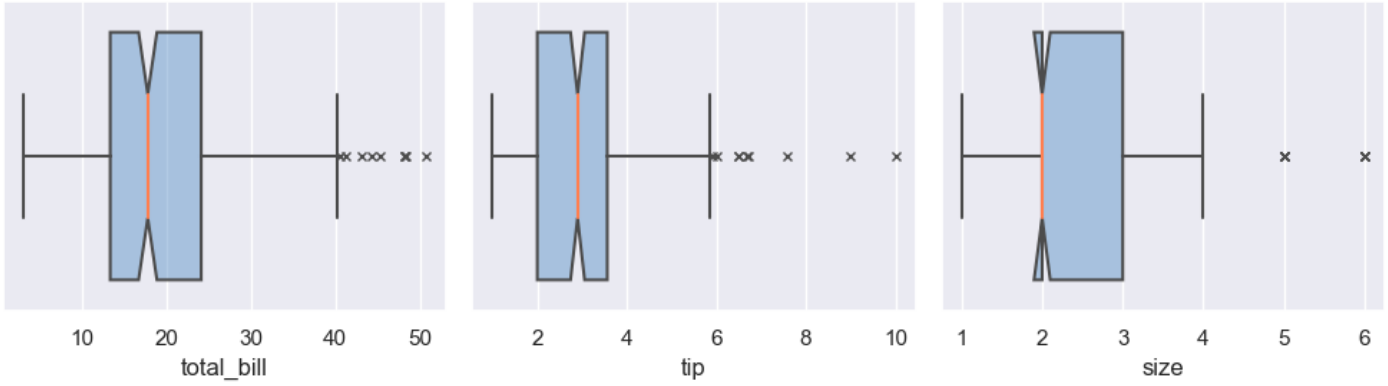
```
<Figure size 1800x600 with 0 Axes>
```



b. Box Plot

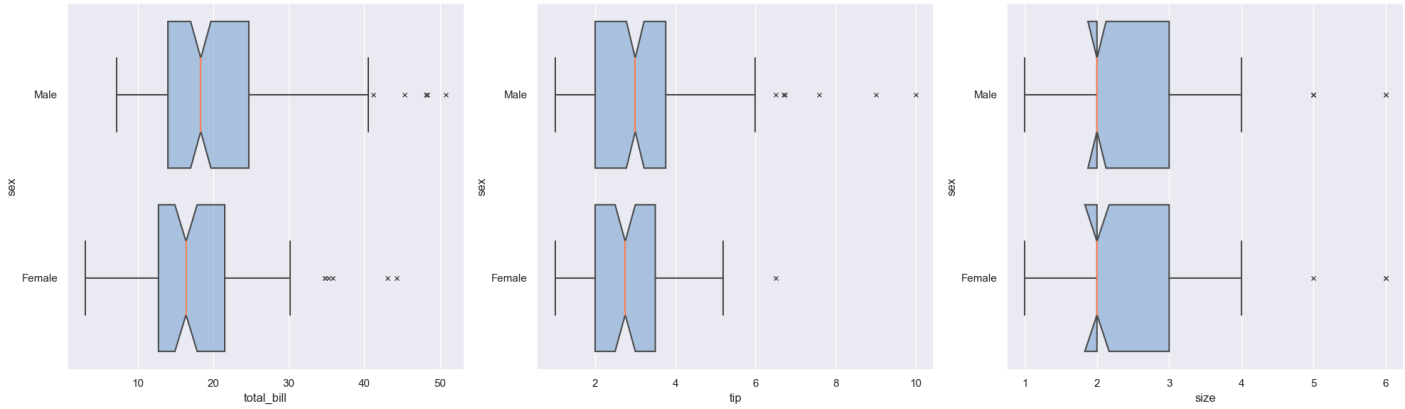
Boxplot grid of all the numerical columns

```
tips_eda.boxplot(height=3, width=10);
```



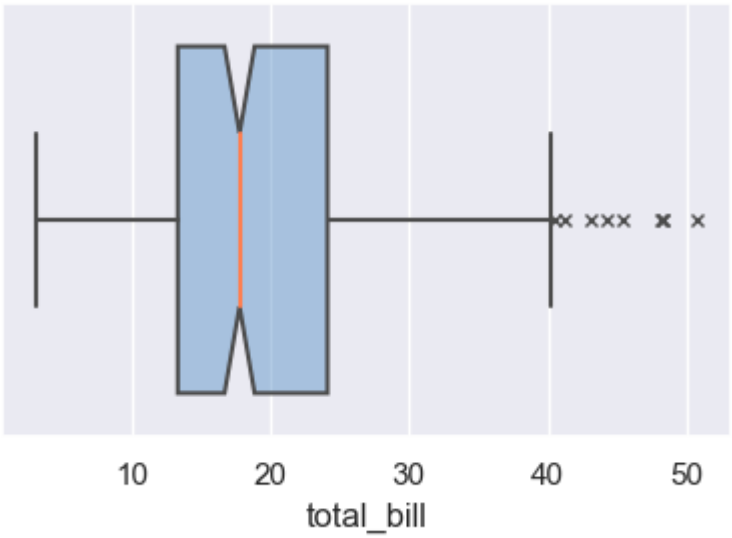
Boxplot grid for all numerical columns with hue specified

```
tips_eda.boxplot(hue='sex', height=6);
```



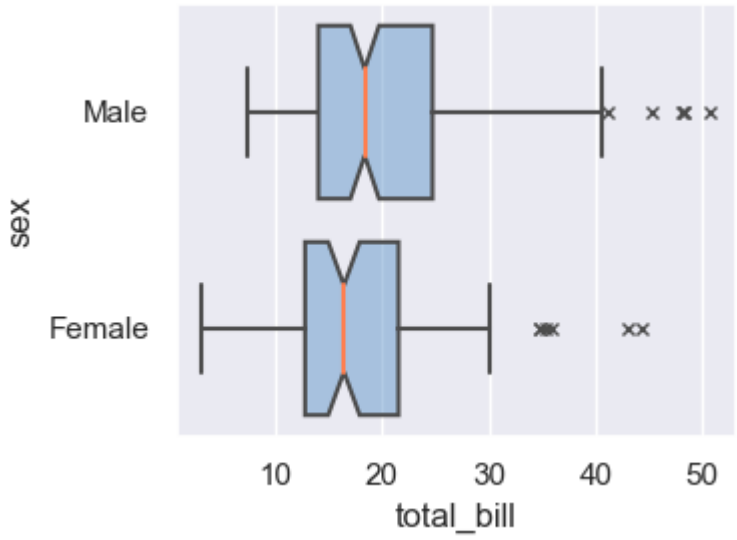
Boxplot for a specific numerical column

```
tips_eda.boxplot(col_to_plot=['total_bill'], width=4, height=3);
```



Boxplot of any numerical with hue specified

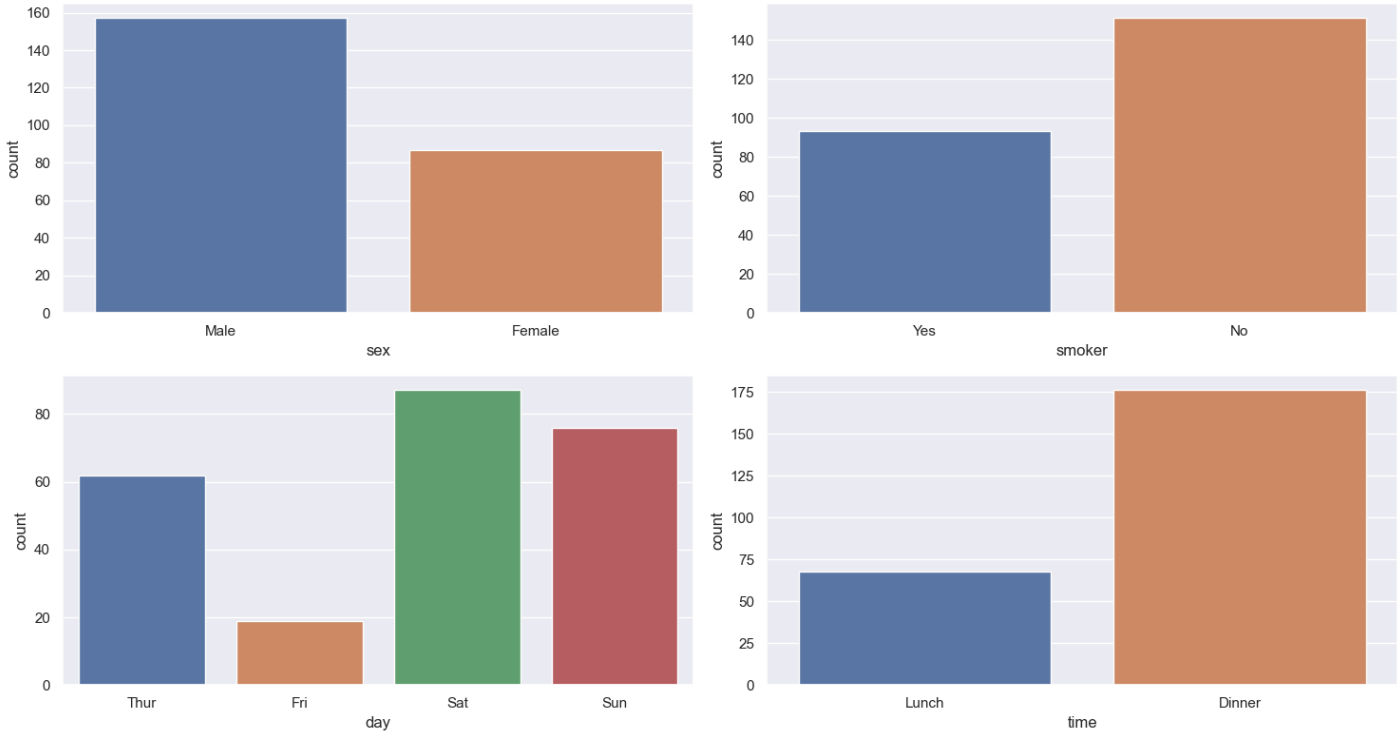
```
tips_eda.boxplot(col_to_plot=['total_bill'], hue='sex', width=4, height=3);
```



c. Count Plot

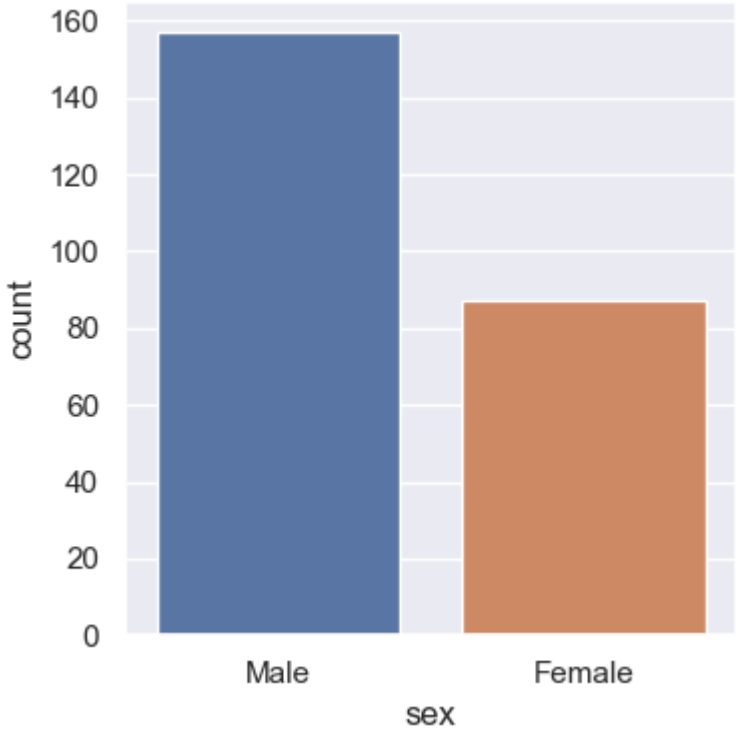
Count plot grid of all the non-numeric columns in data

```
tips_eda.countplot();
```



Count plot of a single categorical column

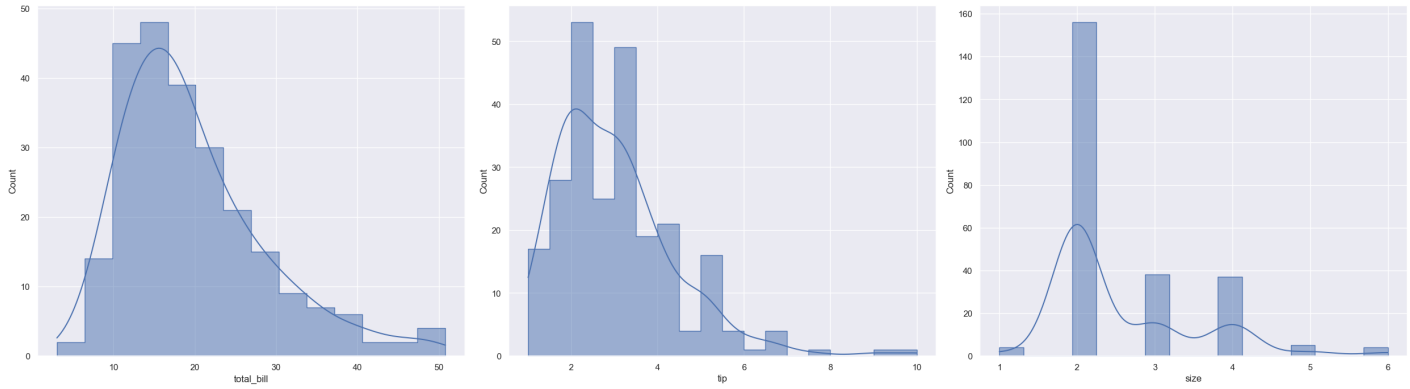
```
tips_eda.countplot(['sex'], height=4, width=4);
```



d. Histogram

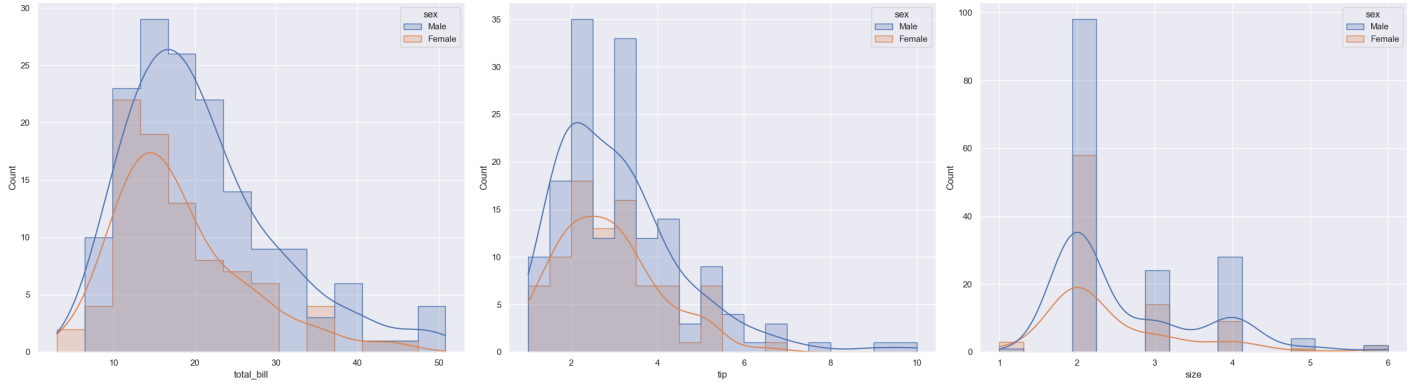
Histogram grid for all the numerical columns

```
tips_eda.histogram(height=7);
```



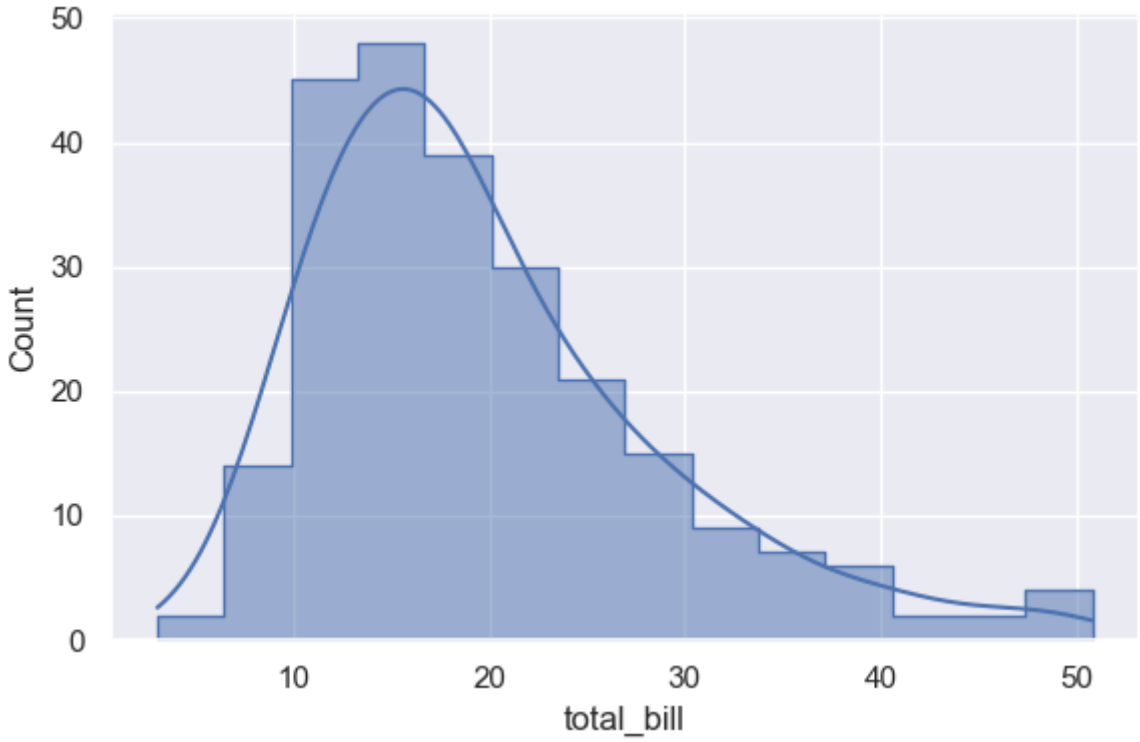
Histogram with a categorical column being passed to hue as the keyword argument

```
tips_eda.histogram(hue = 'sex', height=7);
```



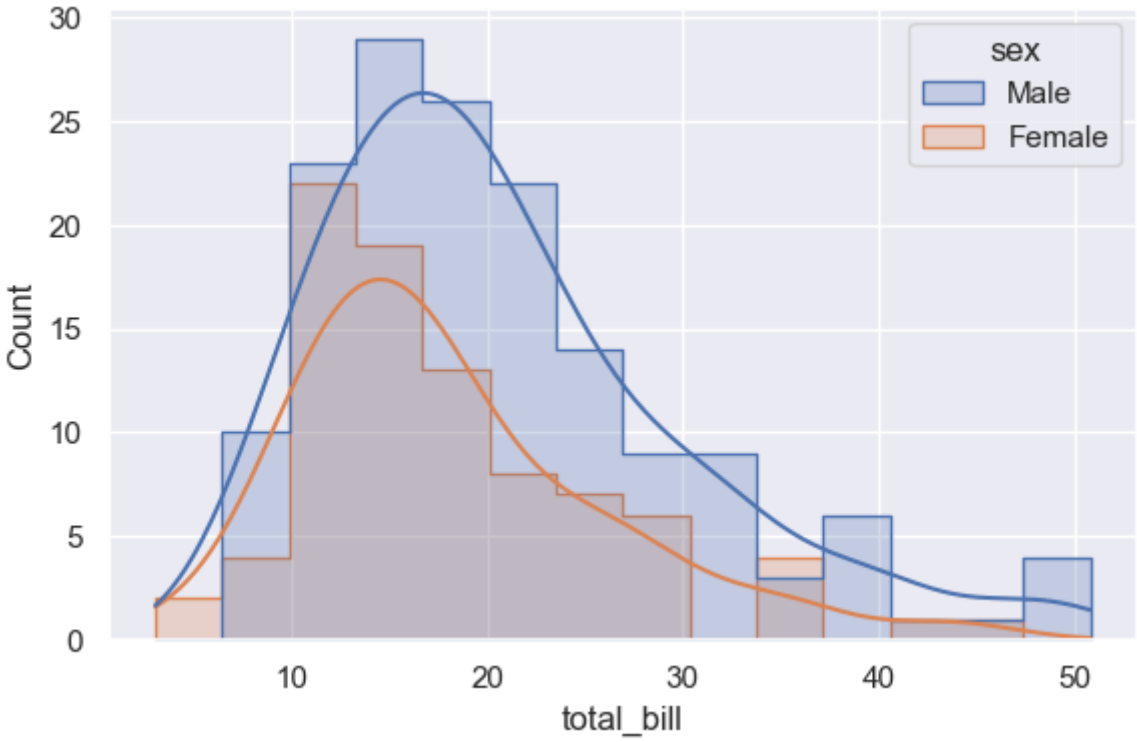
histogram for a specific numerical column

```
tips_eda.histogram(col_to_plot=['total_bill'], height=4, width=6);
```



Histogram for a specific numerical column with hue provided

```
tips_eda.histogram(col_to_plot= ['total_bill'], hue = 'sex', height=4, width=6);
```

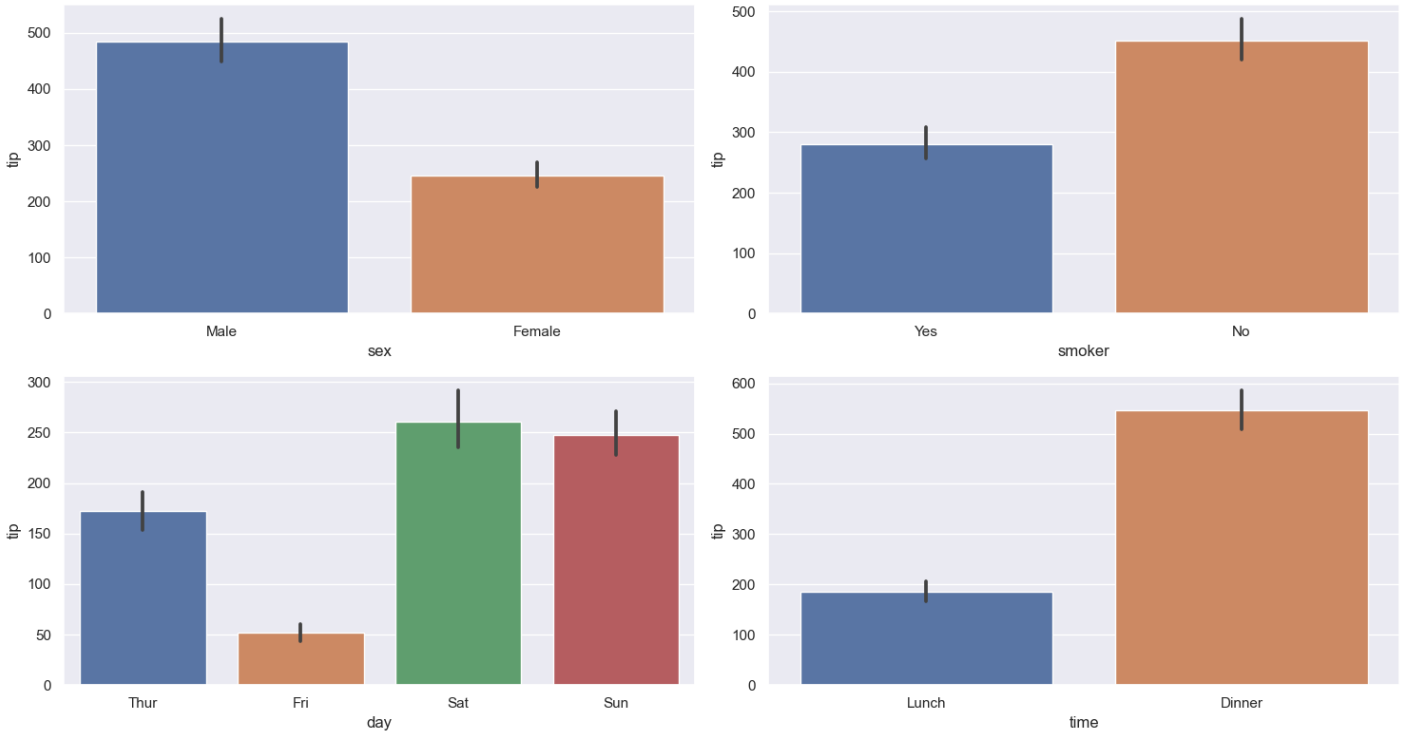


e. Bar chart

Barchart grid of all the categorical columns against a provided numrical column

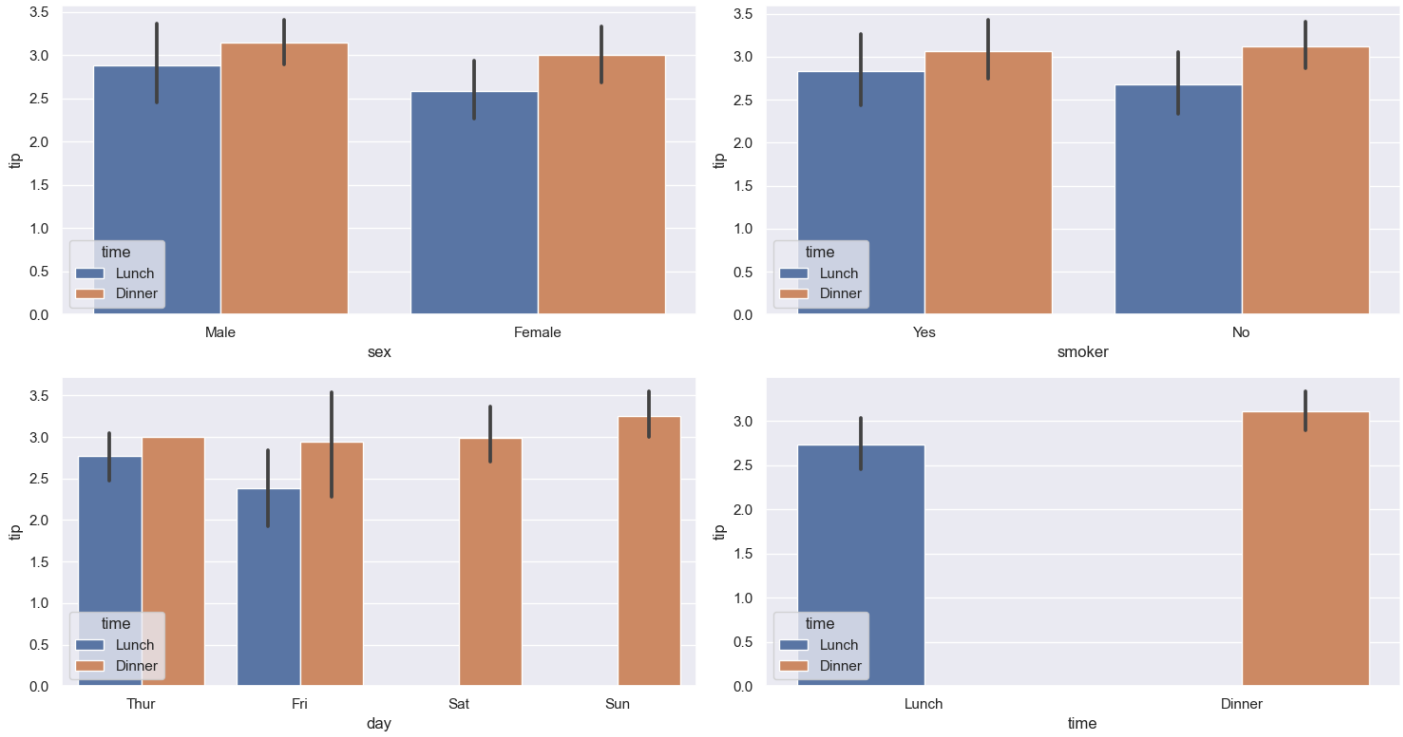
`Note` : Keyword parameter, **estimator** could be any one of **(sum, mean, meadian, std, var)**. Default is set to **mean**

```
tips_eda.barplot(y='tip', estimator='sum');
```



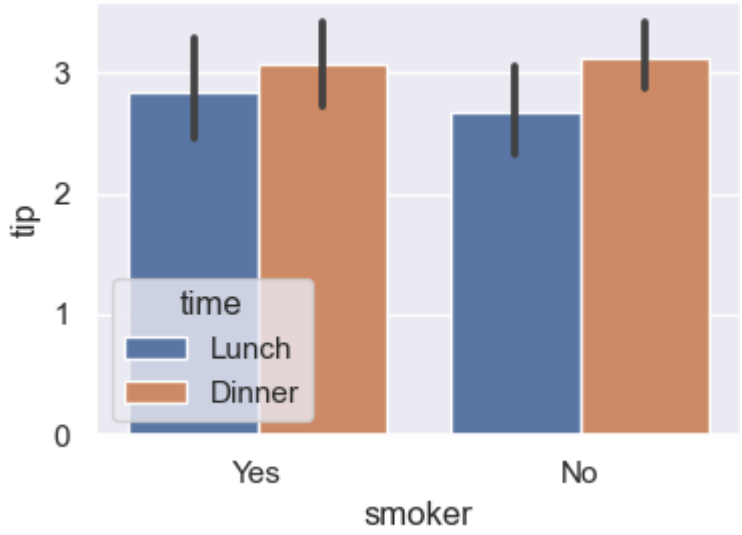
Barchart grid of all the categorical columns against a provided numrical column with hue set to a categorical column

```
tips_eda.barplot(y='tip', hue='time').show()
```



Barchart of single variable selected

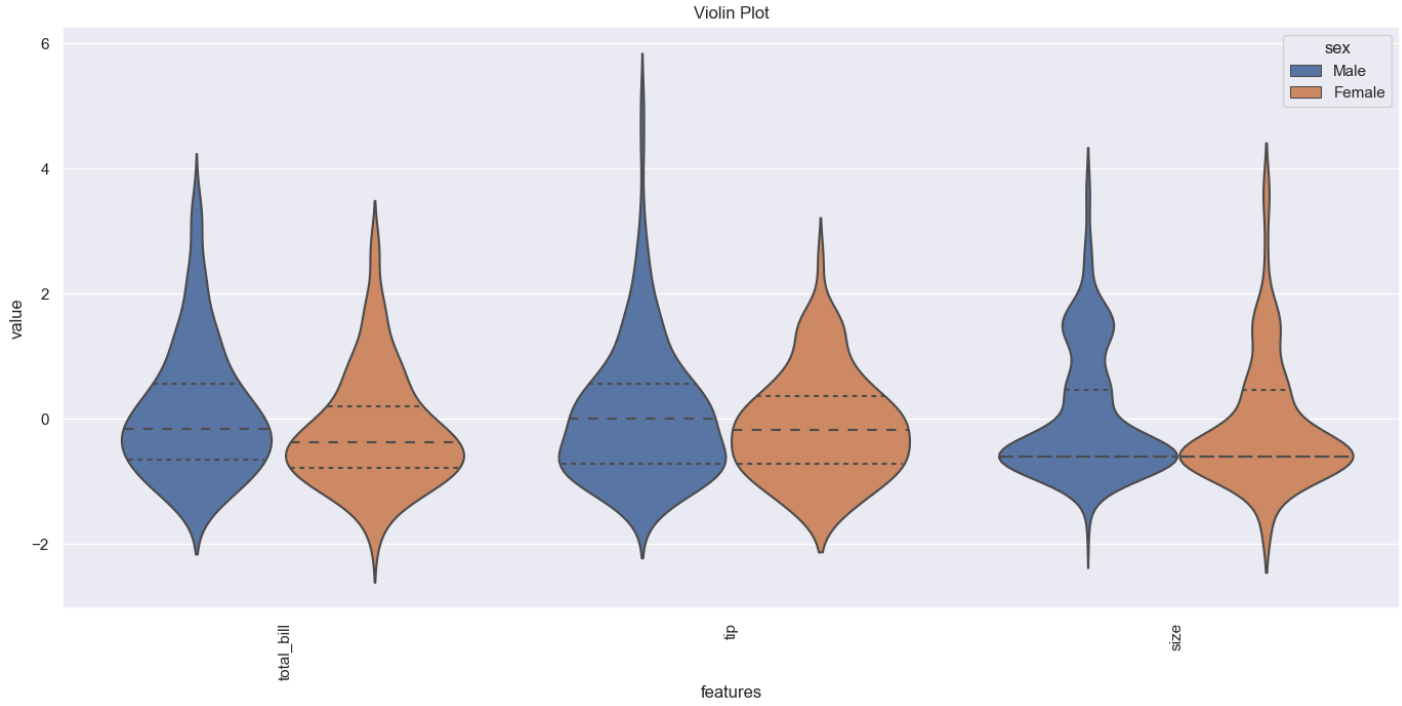
```
tips_eda.barplot(y='tip', col_to_plot='smoker', hue='time', height=3, width=4);
```



f. Violin Plot

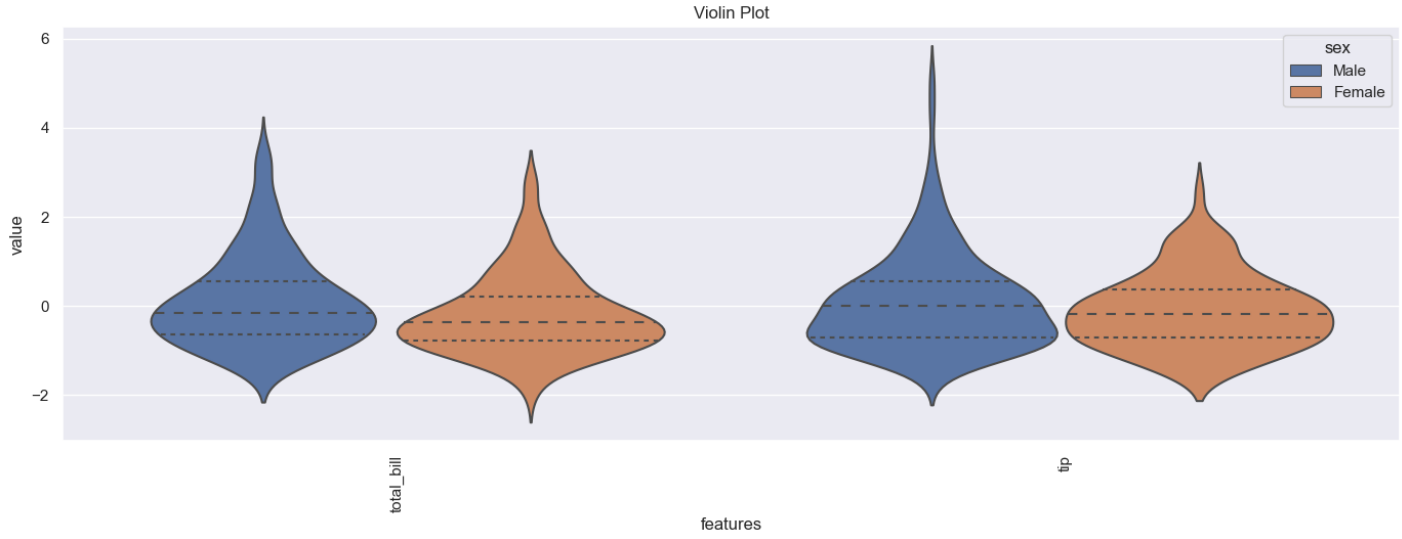
Violinplot for all numerical columns against a target or categorical column pass as keyword argument

```
tips_eda.violinplot(hue='sex', height=7);
```



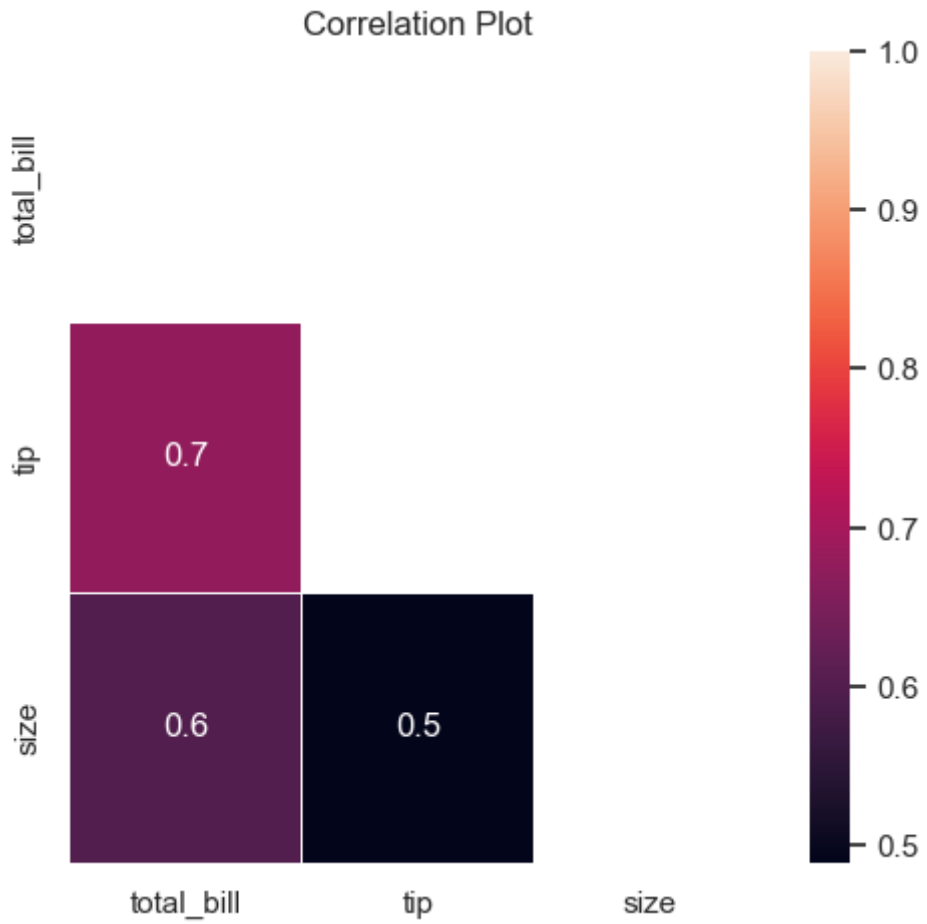
Violinplot for selective numerical columns against a target or categorical column pass as a keyword argument

```
tips_eda.violinplot(col_to_plot= ['total_bill', 'tip'], hue='sex', height=5);
```



g. Correlation Heatmap

```
tips_eda.corr_heatmap(height=5, width=5);
```

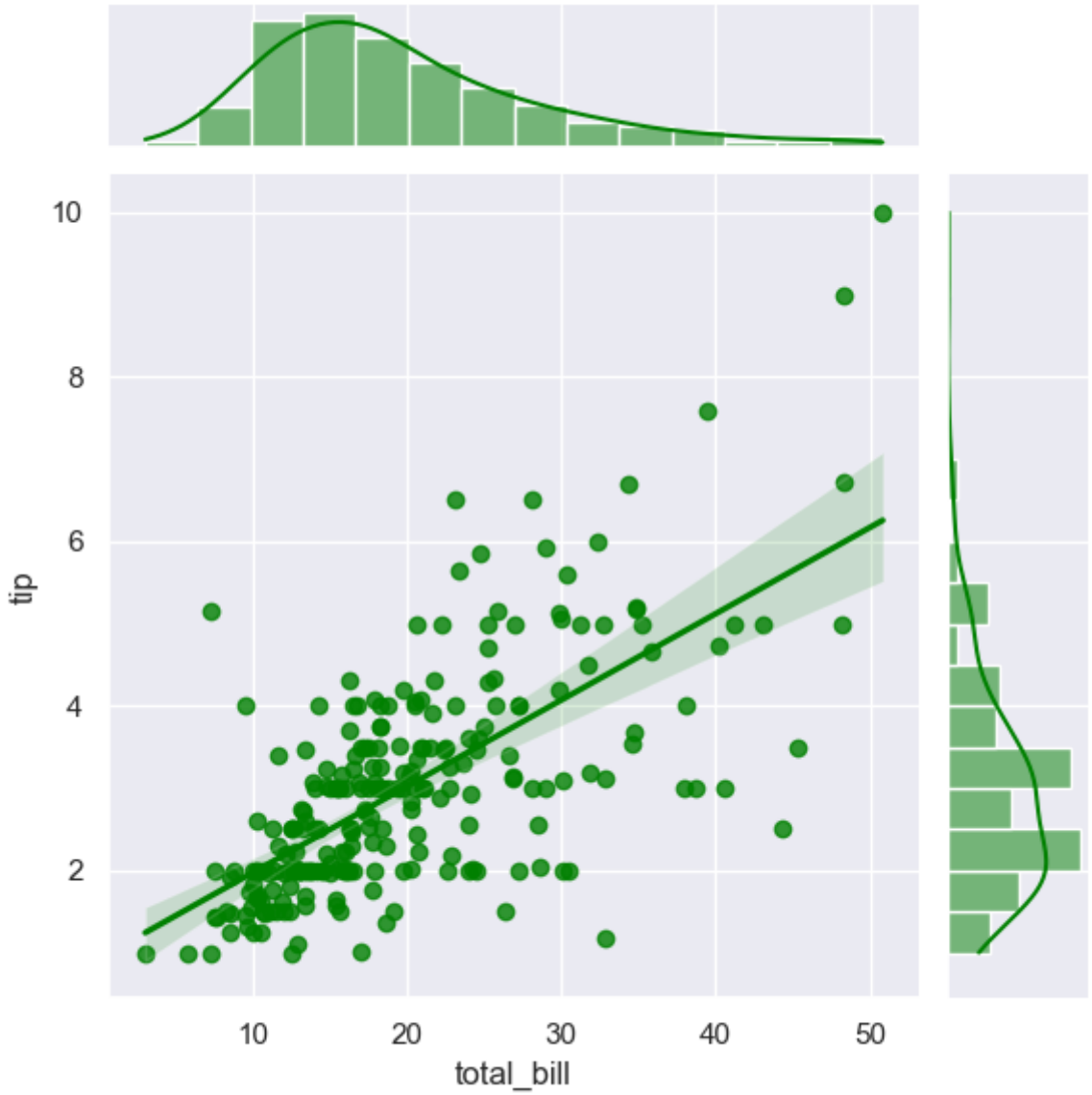


h. Joint Plot

Jointplot of two numerical variables

```
tips_eda.jointplot(x='total_bill',
                   y='tip',
                   height=5, width=5,
                  color = 'green');
```

```
<Figure size 500x500 with 0 Axes>
```

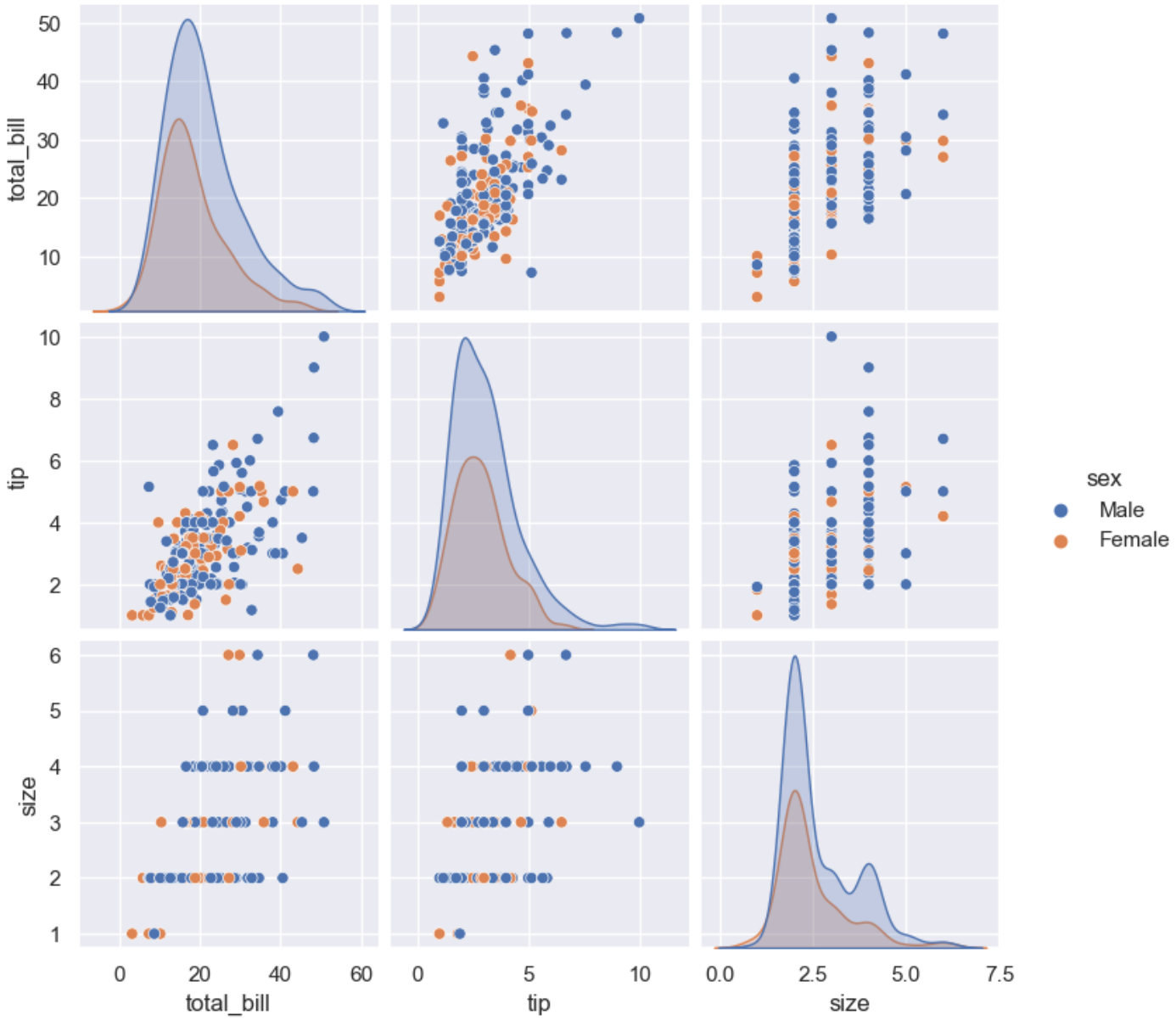


i. Pair Plot

Pairplot of all the numerical features with a categorical column being passed as a keyword argument

```
tips_eda.pairplot(hue='sex', height=5, width=8);
```

```
<Figure size 800x500 with 0 Axes>
```



3. Model Training & Evaluation

Note : This notebook is not focused on model training process so we will NOT deep dive into that as this section is focused on how to visualize and evaluate model performace using vizpool static API

a. Feature Importace

```
# Lets select some columns and try to predict if person is smoker or not
# Getting the feature importance with a sklearn machine learning pipeline
pipeline_data = df[["total_bill", "tip", "size", "sex", "smoker"]]
target_class_labels = pipeline_data['smoker'].unique().tolist()
target = LabelEncoder().fit_transform(pipeline_data.pop('smoker'))
X_train, X_val, y_train, y_val = train_test_split(pipeline_data, target, test_size=0.2,

# Instantiate Evaluation class
model_eval = Evaluation(y_val)

col_trans = ColumnTransformer(transformers=[
    ('num_processing', StandardScaler(),["total_bill", "tip", "size"]),
    ('cat_processing', OneHotEncoder(), ['sex'])
```

```
], remainder= 'drop')
pipe_rfc = Pipeline(steps=[
    ('tranformer', col_trans),
    ('classifier', RandomForestClassifier())
])
pipe_rfc.fit(X_train, y_train)
```

```
Pipeline(steps=[('tranformer',
ColumnTransformer(transformers=[('num_processing',
StandardScaler(),
['total_bill', 'tip',
'size']),
('cat_processing',
OneHotEncoder(), ['sex'])])),
('classifier', RandomForestClassifier())])
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Pipeline

```
Pipeline(steps=[('tranformer',
ColumnTransformer(transformers=[('num_processing',
StandardScaler(),
['total_bill', 'tip',
'size']),
('cat_processing',
OneHotEncoder(), ['sex'])])),
('classifier', RandomForestClassifier())])
```

tranformer: ColumnTransformer

```
ColumnTransformer(transformers=[('num_processing', StandardScaler(),
['total_bill', 'tip', 'size']),
('cat_processing', OneHotEncoder(), ['sex'])])
```

num_processing

```
['total_bill', 'tip', 'size']
```

StandardScaler

```
StandardScaler()
```

cat_processing

```
['sex']
```

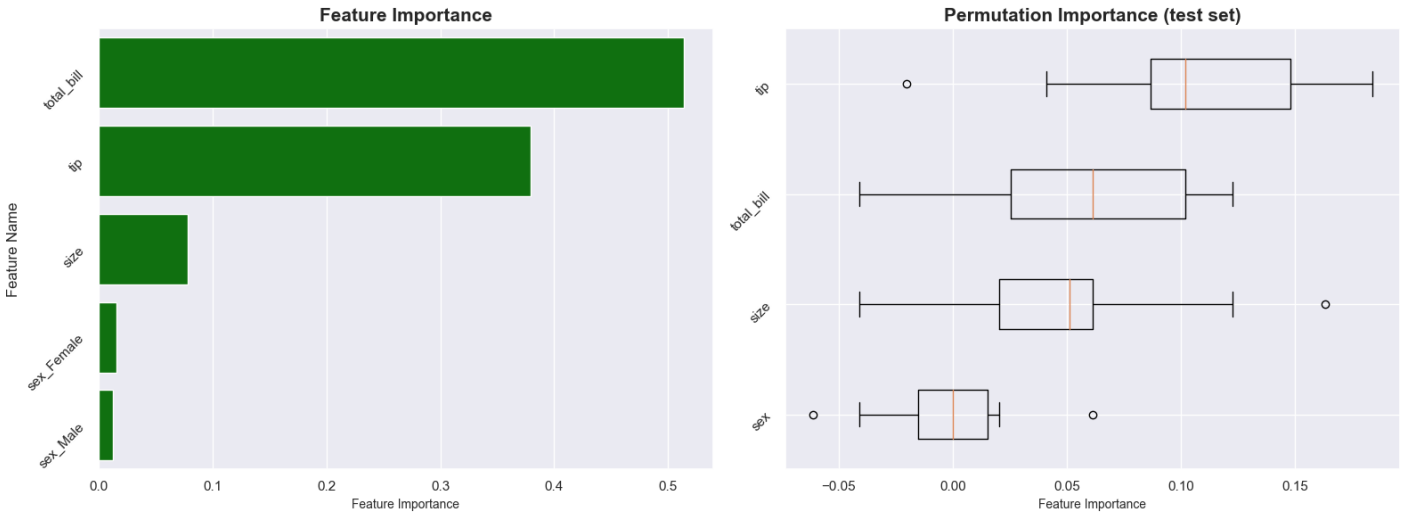
OneHotEncoder

```
OneHotEncoder()
```

RandomForestClassifier

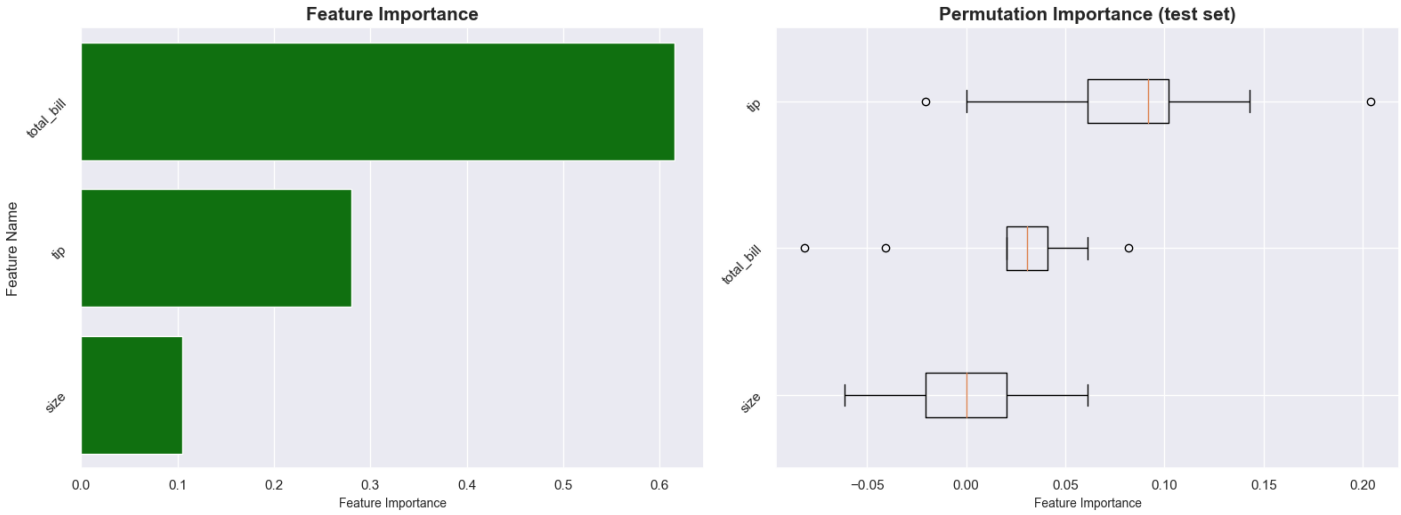
```
RandomForestClassifier()
```

```
model_eval.feature_importance(pipe_rfc, X_val= X_val, pipeline=True,
                              height=6, width = 16);
```



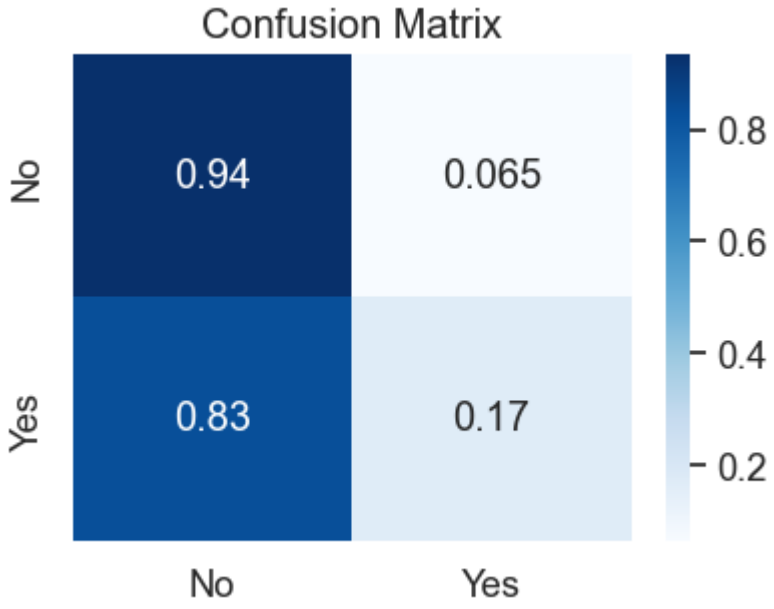

```
# Getting the feature importance with an estimtor
df = df[["total_bill", "tip", "size", "smoker"]]
target_class_labels = df['smoker'].unique().tolist()
target = LabelEncoder().fit_transform(df.pop('smoker'))
X_train, X_val, y_train, y_val = train_test_split(df, target, test_size=0.2, random_sta
X_train.shape, X_val.shape, y_train.shape, y_val.shape

logistic_reg_clf = LogisticRegression()
logistic_reg_clf.fit(X_train, y_train)
logistic_predictions = logistic_reg_clf.predict(X_val)

etc = ExtraTreeClassifier()
etc.fit(X_train, y_train)
etc_predictions = etc.predict(X_val)

svc = SVC(probability=True)
svc.fit(X_train, y_train)
svc_predictions = svc.predict(X_val)

sgd = SGDClassifier()
sgd.fit(X_train, y_train)
sgd_predictions = sgd.predict(X_val)
```

```
plt = model_eval.feature_importance(etc, X_val= X_val,
                              height=6, width = 16);
```

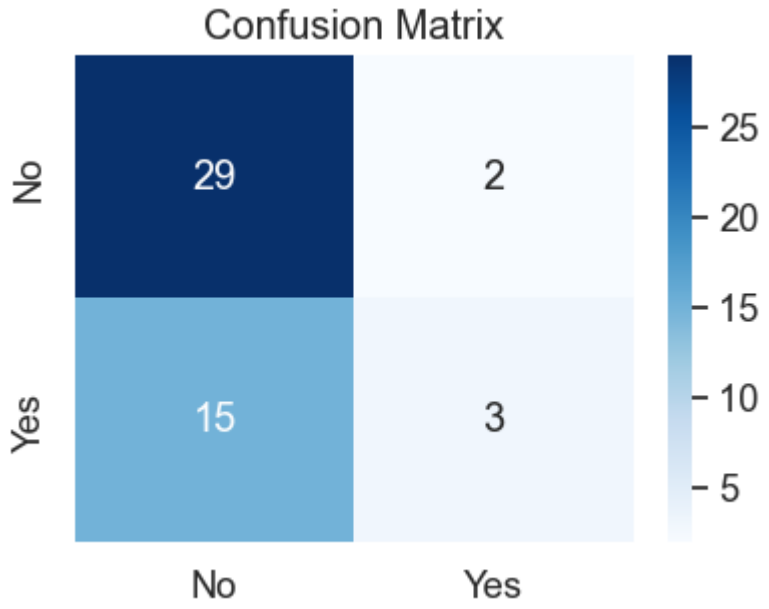


b. Confusion Matrix

```
# Confusion matrix with ratios
model_eval.confusion_matrix(logistic_predictions,
                            target_names=target_class_labels,
                            height=3, width=4);
```

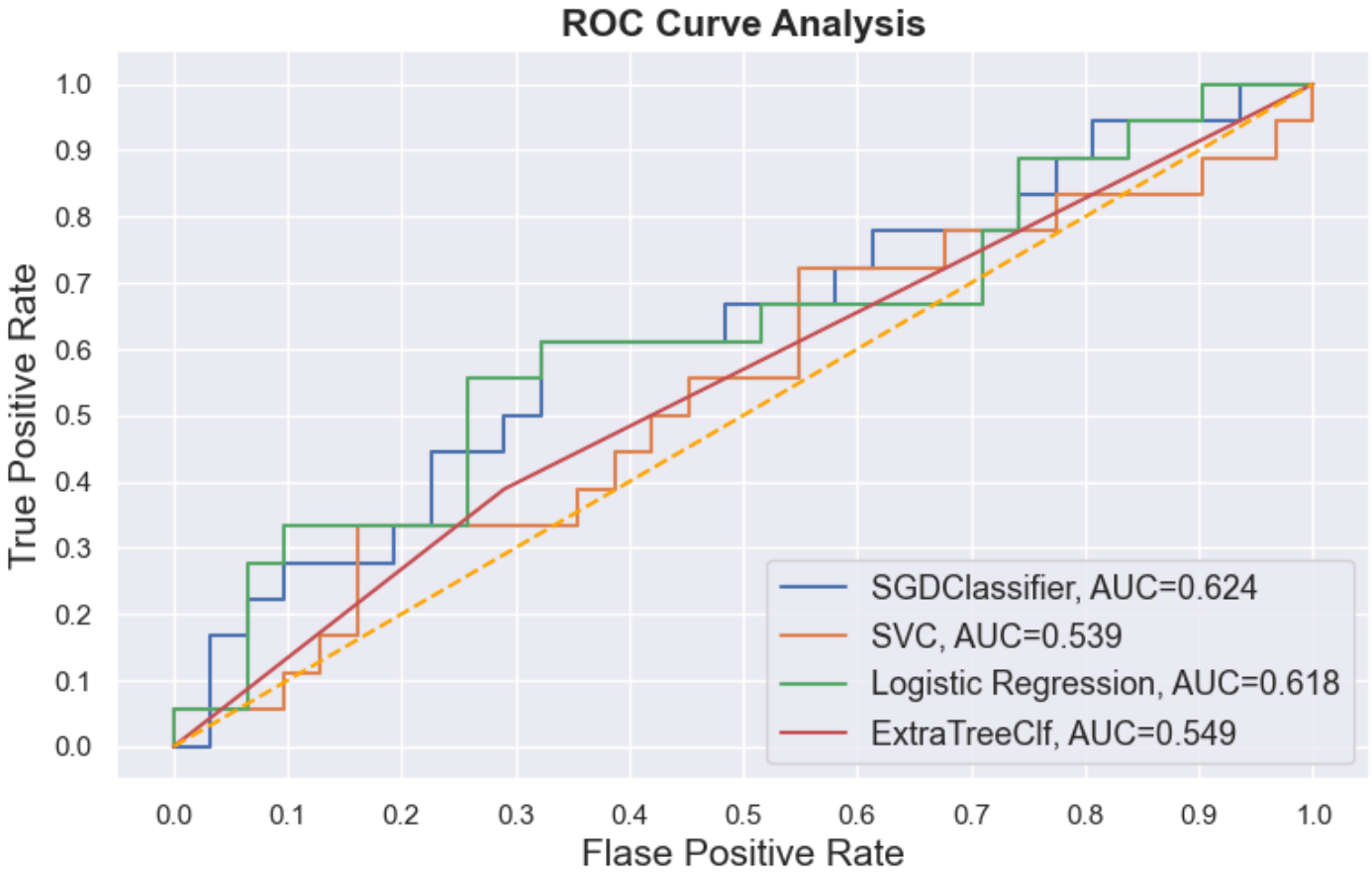



```
# Confusion matrix with Count
model_eval.confusion_matrix(logistic_predictions, target_names=target_class_labels,
                            height=3, width=4, normalize = False);
```



c. AUC ROC Plot

```
model_eval.auc_roc_plot(X_val, [sgd, svc, logistic_reg_clf, etc],
                        ['SGDClassifier', 'SVC', 'Logistic Regression', 'ExtraTreeClf']
                        height=5, width = 8);
```

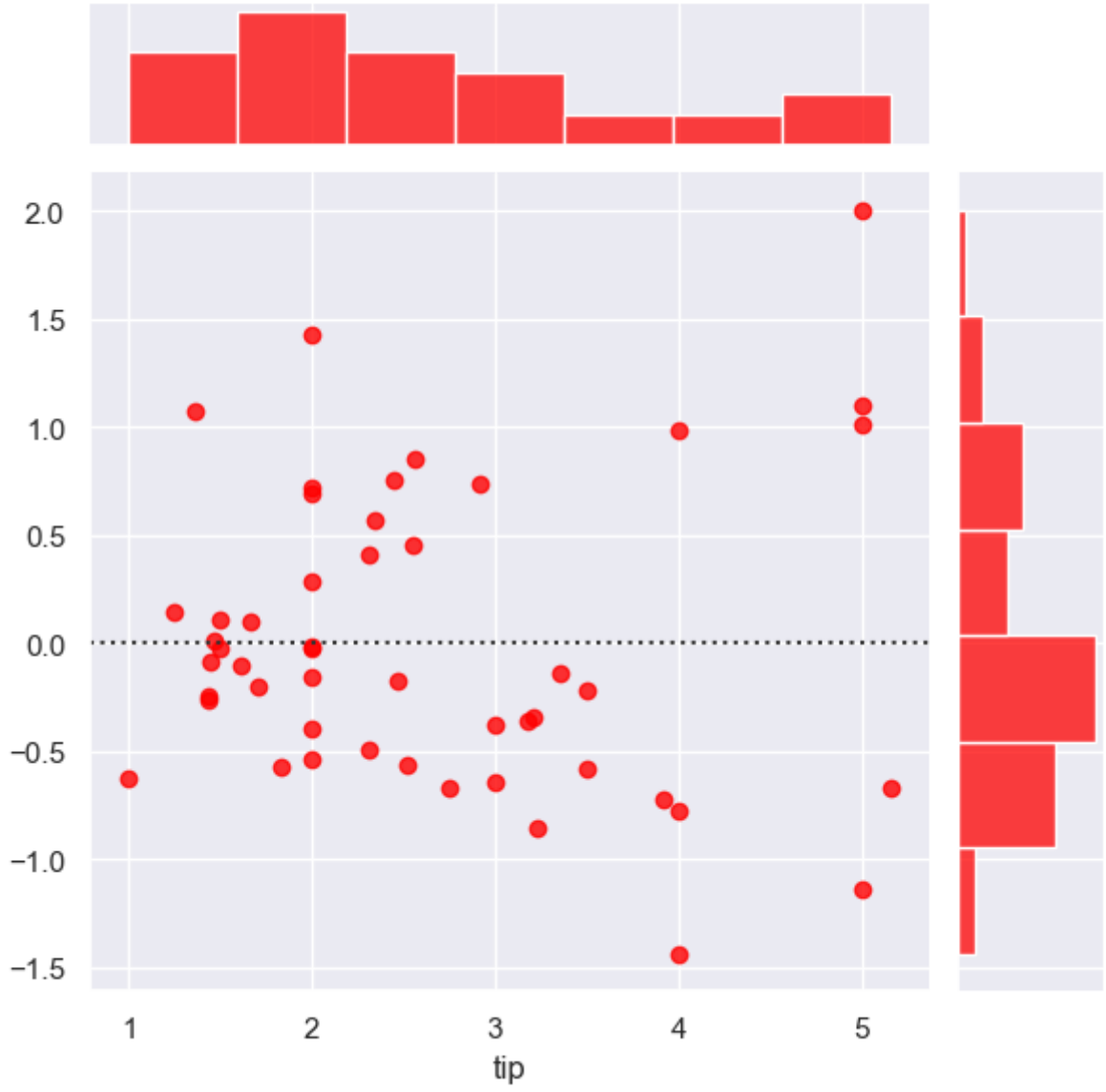


d. Residual Plot

Residual plot is useful for evaluation the results of regression models

```
#Training a Linear Regression model
lr = LinearRegression()
X = df[['size', 'total_bill']]
y = df['tip']
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.2, random_state=42)
lr.fit(X_train, y_train)
y_predicted = lr.predict(X_val)

#Model Evaluation
lr_model_eval = Evaluation(y_val)
lr_model_eval.residplot(y_predicted=y_predicted, color = 'red');
```

```
<Figure size 1800x2000 with 0 Axes>
```



End of Notebook

Let's connect on LinkedIn
Let's connect on GitHub