Summer Capstone-2024- Dhanush Myneni

In [1]:
```
# Importing necessary and needed libraries. Pandas help us to play around and do data manipulation with the dataframes. Seaborn i
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
!pip install xgboost
!pip install lightgbm
from xgboost import XGBRegressor
from lightgbm import LGBMRegressor
```

```
Requirement already satisfied: xgboost in c:\users\dhanu\anaconda3\lib\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\users\dhanu\anaconda3\lib\site-packages (from xgboost) (1.24.3)
Requirement already satisfied: scipy in c:\users\dhanu\anaconda3\lib\site-packages (from xgboost) (1.11.1)
Requirement already satisfied: lightgbm in c:\users\dhanu\anaconda3\lib\site-packages (4.3.0)
Requirement already satisfied: numpy in c:\users\dhanu\anaconda3\lib\site-packages (from lightgbm) (1.24.3)
Requirement already satisfied: scipy in c:\users\dhanu\anaconda3\lib\site-packages (from lightgbm) (1.11.1)
```

In [2]:
```
#Loading the Dataset. Creating a variable named Seattle_permits data to read the csv file.
Seattle_permits_data = pd.read_csv('C:\\Users\\dhanu\\OneDrive\\Desktop\\building-permits.csv')
```

In [3]:
```
Seattle_permits_data.describe() # This describes the numeric data in the dataset by providing key metrics such as the mean, media
```

Out[3]:

	HousingUnits	HousingUnitsRemoved	HousingUnitsAdded	EstProjectCost	OriginalZip	Latitude	Longitude
count	822.000000	54067.000000	54067.000000	8.196400e+04	113861.000000	115134.000000	1.151340e+05
mean	1.021898	0.173581	1.879464	5.537356e+05	98119.985763	399.209183	1.808644e+03
std	6.846464	1.904947	16.510370	5.031794e+06	21.574942	9050.706818	4.949283e+04
min	0.000000	0.000000	0.000000	0.000000e+00	98004.000000	47.495829	-1.224304e+02
25%	0.000000	0.000000	0.000000	1.500000e+04	98106.000000	47.584450	-1.223626e+02
50%	0.000000	0.000000	0.000000	5.000000e+04	98115.000000	47.626412	-1.223336e+02
75%	1.000000	0.000000	1.000000	2.182250e+05	98122.000000	47.668965	-1.223044e+02
max	161.000000	272.000000	1097.000000	2.922400e+08	98199.000000	269787.042000	1.290810e+06

In [4]:
```
Seattle_permits_data.info() #The dataset has 117524 rows and 24 attributes.
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117524 entries, 0 to 117523
Data columns (total 24 columns):
 #   Column                 Non-Null Count   Dtype  
---  ------                 --------------   -----  
 0   PermitNum              117524 non-null  object 
 1   PermitClass            111208 non-null  object 
 2   PermitClassMapped      111208 non-null  object 
 3   PermitTypeMapped       117524 non-null  object 
 4   PermitTypeDesc         111975 non-null  object 
 5   Description            117410 non-null  object 
 6   HousingUnits           822 non-null     float64
 7   HousingUnitsRemoved    54067 non-null   float64
 8   HousingUnitsAdded      54067 non-null   float64
 9   EstProjectCost         81964 non-null   float64
 10  AppliedDate            105202 non-null  object 
 11  IssuedDate             87195 non-null   object 
 12  ExpiresDate            87308 non-null   object 
 13  CompletedDate          65850 non-null   object 
 14  StatusCurrent          117524 non-null  object 
 15  OriginalAddress1       117466 non-null  object 
 16  OriginalCity           117247 non-null  object 
 17  OriginalState          117112 non-null  object 
 18  OriginalZip            113861 non-null  float64
 19  ContractorCompanyName  26596 non-null   object 
 20  Link                   117524 non-null  object 
 21  Latitude               115134 non-null  float64
 22  Longitude              115134 non-null  float64
 23  Location_1             117466 non-null  object 
dtypes: float64(7), object(17)
memory usage: 21.5+ MB
```

In [5]:
```python
Seattle_permits_data.head(10) #The head function gives a quick overview of how our data looks like.
```

Out[5]:

	PermitNum	PermitClass	PermitClassMapped	PermitTypeMapped	PermitTypeDesc	Description	HousingUnits	HousingUnitsRemoved	HousingUnitsAdded
0	6315561-BK	Commercial	Non-Residential	Building	Tenant Improvement Pre-Approval	Blanket Permit for interior non-structural alt...	NaN	NaN	NaN
1	6502207-DM	Commercial	Non-Residential	Demolition	Demolition	Demolish existing gas station, per plan	NaN	0.0	0.0
2	6525734-CN	Single Family/Duplex	Residential	Building	Addition/Alteration	Construct addition and substantial alterations...	NaN	NaN	NaN
3	6542481-DM	Commercial	Non-Residential	Demolition	Demolition	Demolition of commercial building	NaN	NaN	NaN
4	6587701-CN	Single Family/Duplex	Residential	Building	Addition/Alteration	Existing, is a single-family residence and car...	0.0	0.0	0.0
5	6591977-DM	Single Family/Duplex	Residential	Demolition	Demolition	*STFI* Demolish existing duplex, subject to fi...	NaN	2.0	0.0
6	6592354-DM	Multifamily	Residential	Demolition	Demolition	*STFI*Subject to field inspection demolish exi...	NaN	1.0	0.0
7	6592456-DM	Single Family/Duplex	Residential	Demolition	Demolition	Demolish existing SFR per (STFI) Subject to Fi...	NaN	1.0	0.0
8	6595839-CN	Multifamily	Residential	Building	NONE	Establish use for the record as a triplex (194...	2.0	0.0	1.0
9	6602287-DM	Multifamily	Residential	Demolition	Demolition	Demolish existing SFR per STFI, Subject to Fie...	NaN	1.0	0.0

10 rows × 24 columns

In [6]:
```python
Seattle_permits_data.dropna(axis=0) #This drops any rows where there are missing values or na values.
```

Out[6]:

PermitNum	PermitClass	PermitClassMapped	PermitTypeMapped	PermitTypeDesc	Description	HousingUnits	HousingUnitsRemoved	HousingUnitsAdded	EstI

0 rows × 24 columns

In [7]:
```python
# Data Preprocessing
# Check for necessary columns
Columns_required = ['AppliedDate', 'IssuedDate']
missing_columns = [col for col in Columns_required if col not in Seattle_permits_data.columns]
if missing_columns:
    raise ValueError(f"Missing required columns: {missing_columns}")
```

In [8]:
```python
# Convert date columns to datetime
Seattle_permits_data['AppliedDate'] = pd.to_datetime(Seattle_permits_data['AppliedDate'], errors='coerce')
Seattle_permits_data['IssuedDate'] = pd.to_datetime(Seattle_permits_data['IssuedDate'], errors='coerce')
```

In [9]:
```python
#Calculating our target variable ApprovalTime which is the difference of days between the issued date and applied date.

Seattle_permits_data['ApprovalTime'] = (Seattle_permits_data['IssuedDate'] - Seattle_permits_data['AppliedDate']).dt.days
```

In [10]:
```python
# Drop rows with missing ApprovalTime
Seattle_permits_data = Seattle_permits_data.dropna(subset=['ApprovalTime'])
```

In [11]:
```python
#dropping columns in which the na values cross 60% of the total length of the the data.
Seattle_permits_data = Seattle_permits_data.dropna(axis=1, thresh=len(Seattle_permits_data) * 0.6)
```

In [12]:
```python
Seattle_permits_data.info() # Data at our disposal after dropping the rows with na values and columns with too many na values. Th
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 87185 entries, 1 to 117512
Data columns (total 21 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   PermitNum          87185 non-null  object
 1   PermitClass        83834 non-null  object
 2   PermitClassMapped  83834 non-null  object
 3   PermitTypeMapped   87185 non-null  object
 4   PermitTypeDesc     84535 non-null  object
 5   Description        87183 non-null  object
 6   EstProjectCost     75541 non-null  float64
 7   AppliedDate        87185 non-null  datetime64[ns]
 8   IssuedDate         87185 non-null  datetime64[ns]
 9   ExpiresDate        87170 non-null  object
 10  CompletedDate      65847 non-null  object
 11  StatusCurrent      87185 non-null  object
 12  OriginalAddress1   87172 non-null  object
 13  OriginalCity       87105 non-null  object
 14  OriginalState      87085 non-null  object
 15  OriginalZip        84358 non-null  float64
 16  Link               87185 non-null  object
 17  Latitude           85156 non-null  float64
 18  Longitude          85156 non-null  float64
 19  Location_1         87172 non-null  object
 20  ApprovalTime       87185 non-null  float64
dtypes: datetime64[ns](2), float64(5), object(14)
memory usage: 14.6+ MB
```

In [13]:
```python
# Handle missing values in other columns (e.g., filling with median or mode)
for col in Seattle_permits_data.columns:
    if Seattle_permits_data[col].dtype == 'object':
        Seattle_permits_data[col] = Seattle_permits_data[col].fillna(Seattle_permits_data[col].mode()[0])
    else:
        Seattle_permits_data[col] = Seattle_permits_data[col].fillna(Seattle_permits_data[col].median())
```

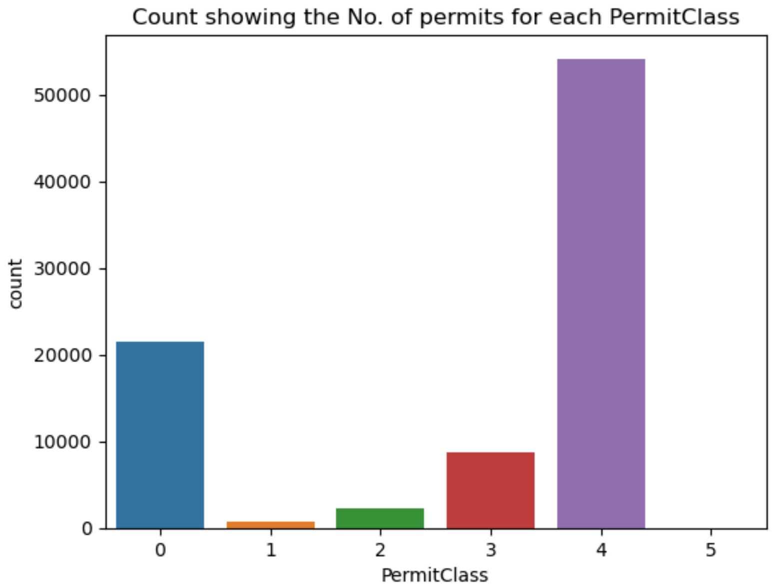
In [14]:
```python
# Reduce cardinality of high-cardinality categorical variables
# We'll encode categorical features with high cardinality using Label Encoding
label_encoders = {}
categorical_features = Seattle_permits_data.select_dtypes(include=['object', 'category']).columns

for col in categorical_features:
    label_encoders[col] = LabelEncoder()
    Seattle_permits_data[col] = label_encoders[col].fit_transform(Seattle_permits_data[col])
```

In [15]:
```python

Seattle_permits_data['Year'] = pd.DatetimeIndex(Seattle_permits_data['AppliedDate']).year
Seattle_permits_data['Month'] = pd.DatetimeIndex(Seattle_permits_data['AppliedDate']).month
```

Exploratory Data Analysis

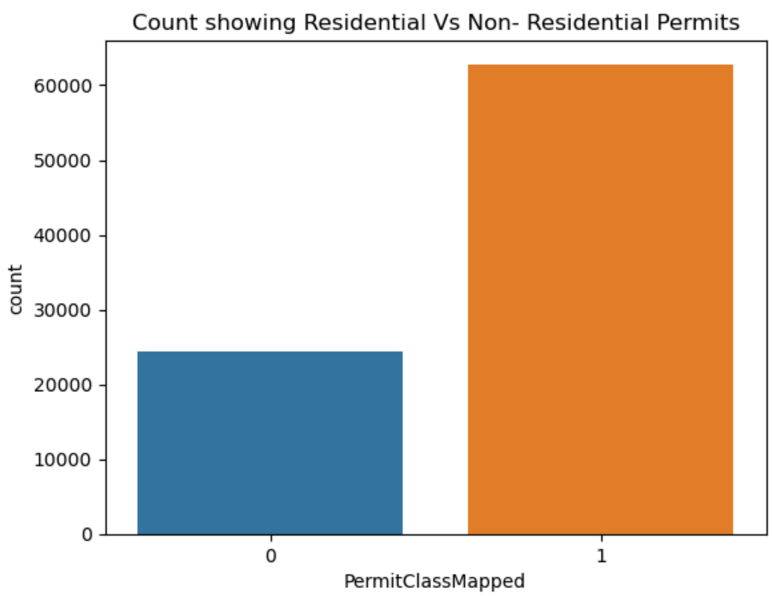
In [16]:
```
sns.countplot(x='PermitClass', data=Seattle_permits_data)
plt.title('Count showing the No. of permits for each PermitClass')
```

Out[16]: `Text(0.5, 1.0, 'Count showing the No. of permits for each PermitClass')`



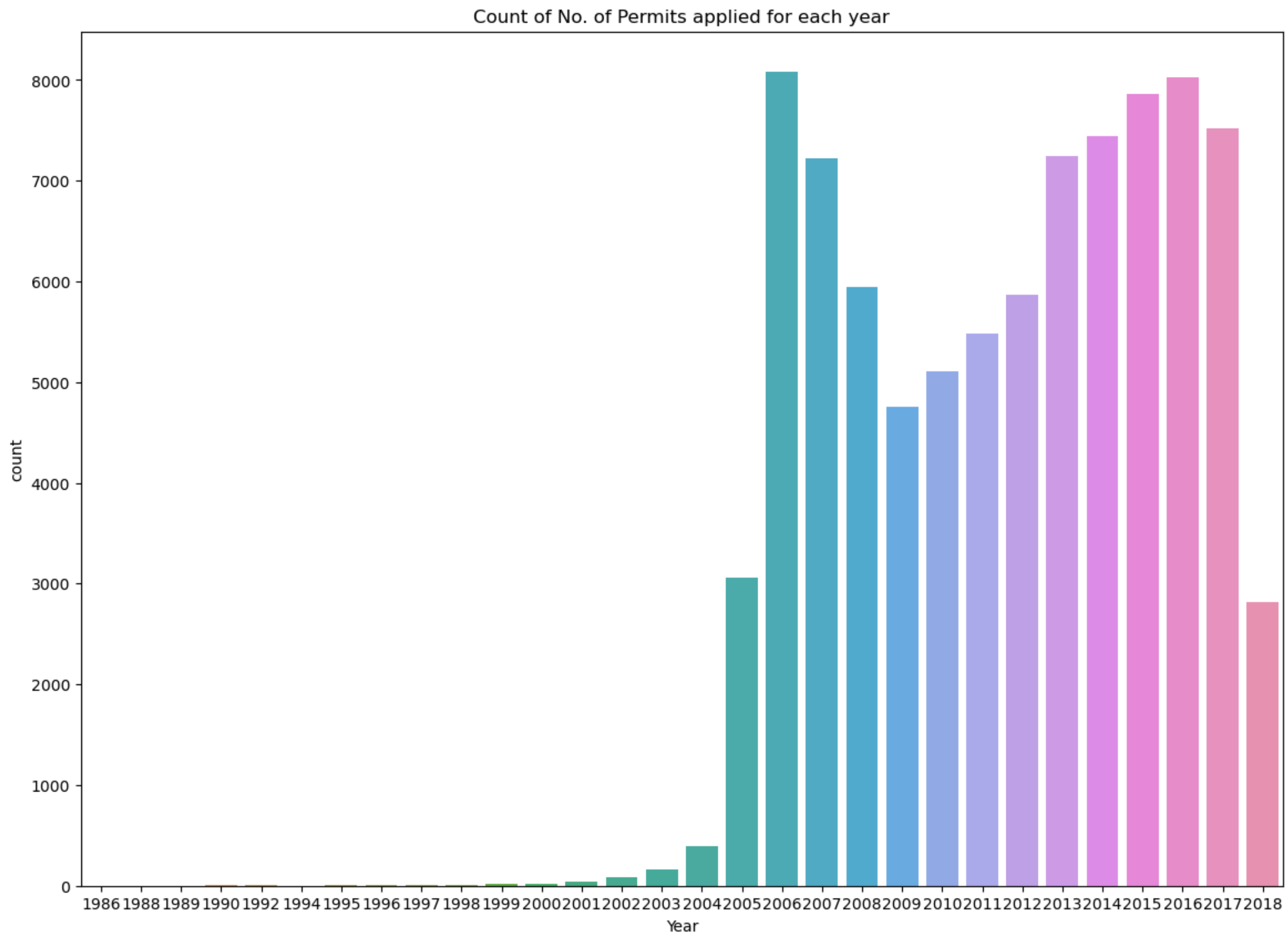
In [17]:
```
sns.countplot(x='PermitClassMapped', data=Seattle_permits_data)
plt.title('Count showing Residential Vs Non- Residential Permits')
```

Out[17]: `Text(0.5, 1.0, 'Count showing Residential Vs Non- Residential Permits')`

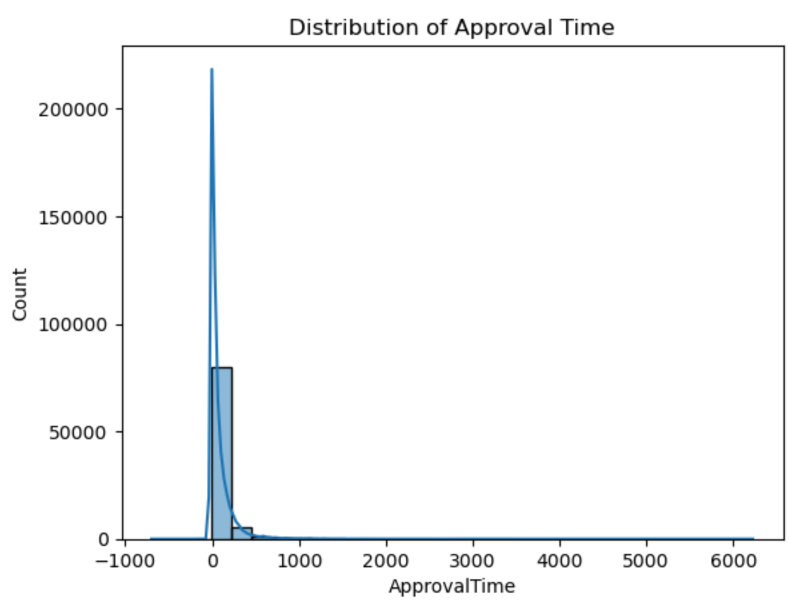


In [18]:
```
plt.figure(figsize=(14, 10))
sns.countplot(x='Year', data=Seattle_permits_data)
plt.title('Count of No. of Permits applied for each year')
```

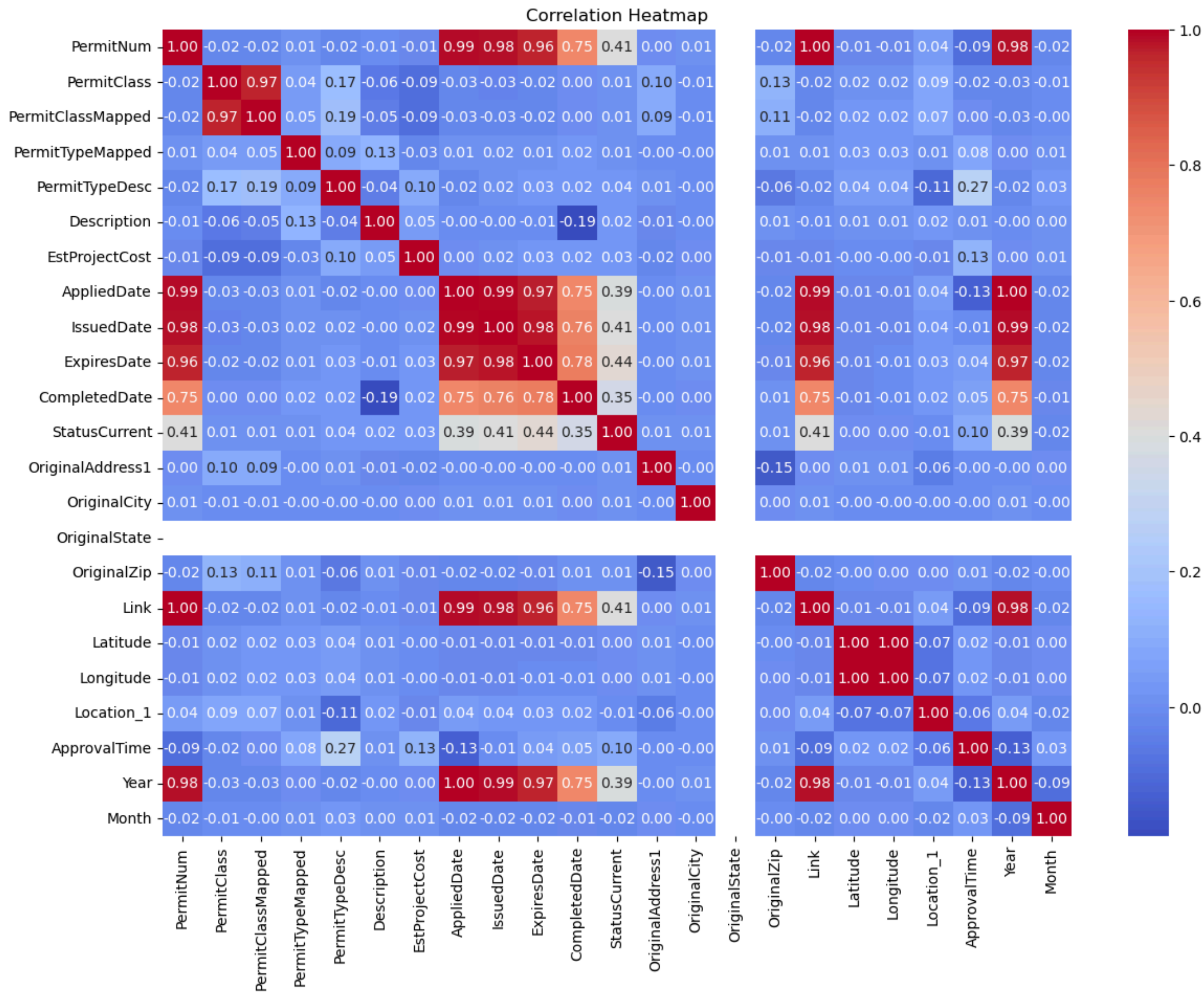
Out[18]: Text(0.5, 1.0, 'Count of No. of Permits applied for each year')



In [19]:
```
# Plotting the distribution of ApprovalTime
sns.histplot(Seattle_permits_data['ApprovalTime'], bins=30, kde=True)
plt.title('Distribution of Approval Time')
plt.show(block=True)
```

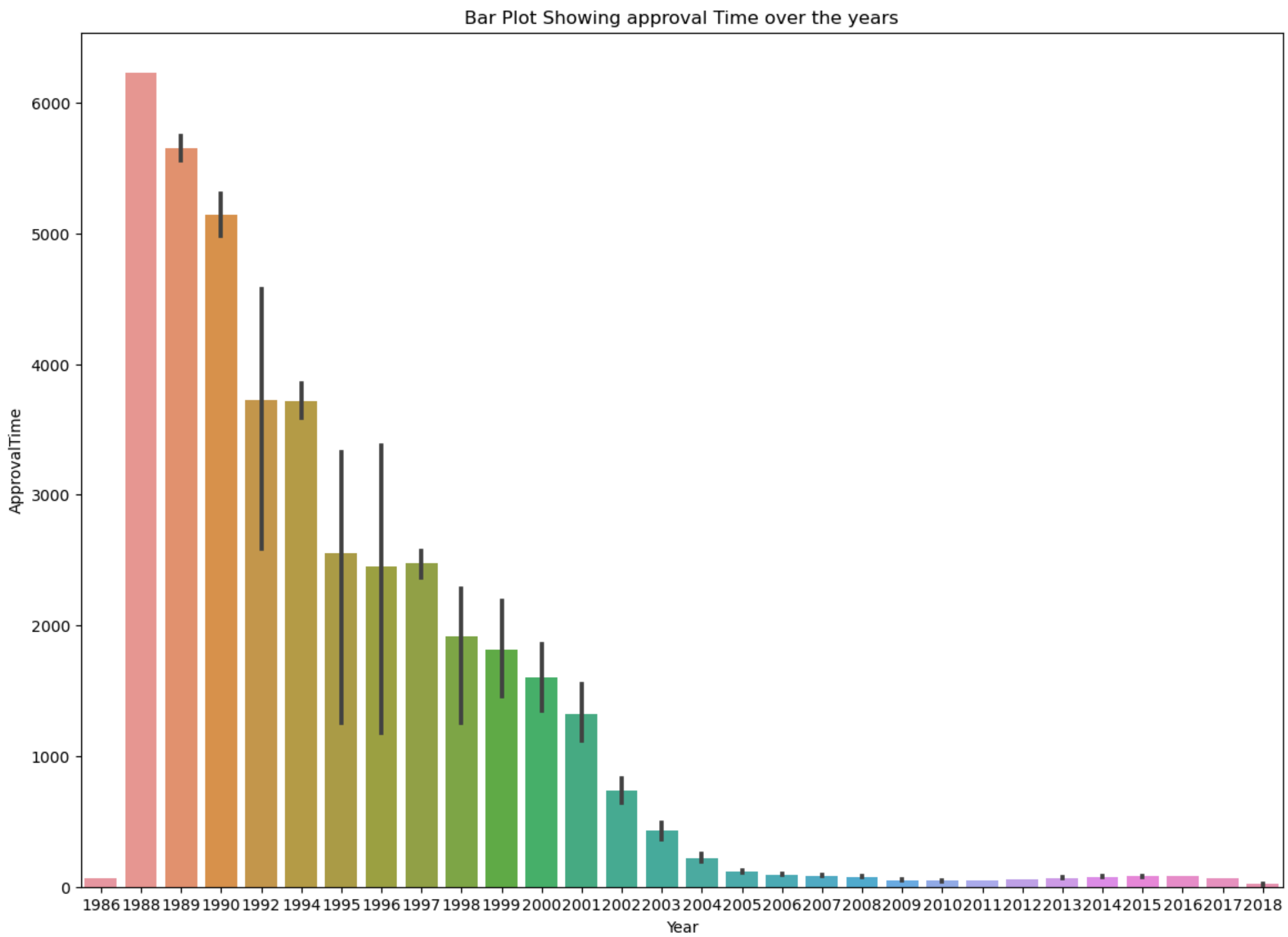


In [20]:
```
plt.figure(figsize=(14, 10))
sns.heatmap(Seattle_permits_data.corr(), annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show(block=True)
```



In [21]:
```python
plt.figure(figsize=(14, 10))
sns.barplot(x= 'Year', y= 'ApprovalTime', data=Seattle_permits_data)
plt.title("Bar Plot Showing approval Time over the years")

#It could be noticed that over the years the approval time has drastically come down over the years
```

Out[21]:
```
Text(0.5, 1.0, 'Bar Plot Showing approval Time over the years')
```

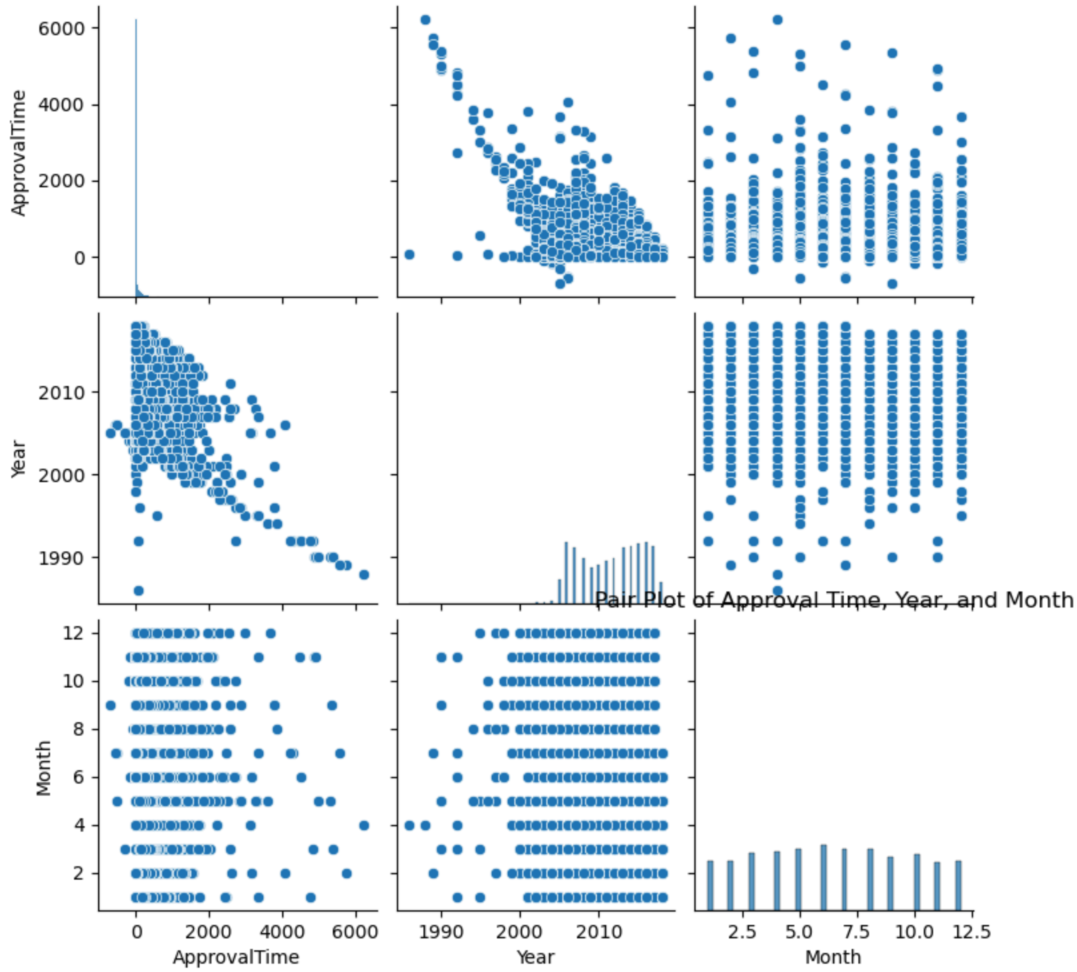


In [22]:
```
plt.figure(figsize=(12, 10))
sns.pairplot(Seattle_permits_data[['ApprovalTime', 'Year', 'Month']])
plt.title('Pair Plot of Approval Time, Year, and Month')
```

```
C:\Users\dhanu\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

Out[22]:
```
Text(0.5, 1.0, 'Pair Plot of Approval Time, Year, and Month')
```

```
<Figure size 1200x1000 with 0 Axes>
```



In [23]:
```
#Defining target variable and the independent variable
X = Seattle_permits_data.drop(columns=['ApprovalTime', 'AppliedDate', 'IssuedDate'], axis=1)
y = Seattle_permits_data['ApprovalTime']
```

In [24]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Training the models

Linear Regression

In [25]:
```
#Create an instance of a linear regression model

lm= LinearRegression()
```

In [26]:
```
#fiitting the model
lm.fit(X_train, y_train)
```

Out[26]:
```
▾ LinearRegression
LinearRegression()
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.linear_model.LinearRegression.html)

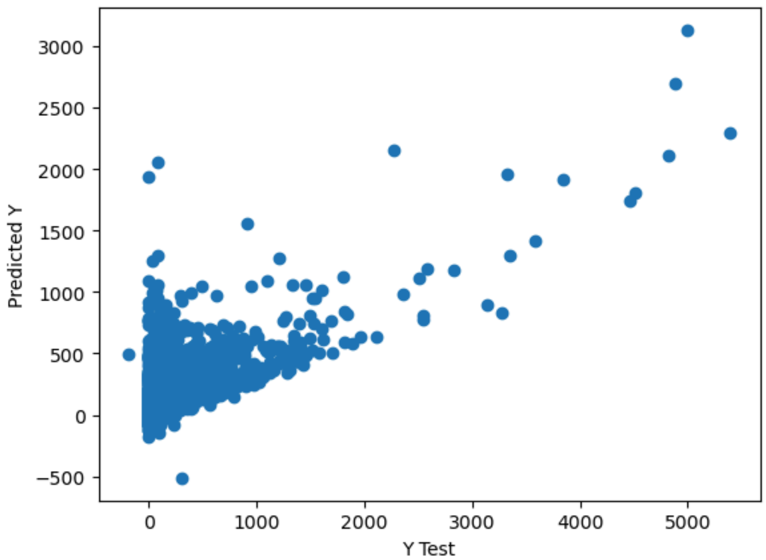
In [27]:
```
print('Coefficients: \n', lm.coef_)
```

```
Coefficients:
 [ 1.74145721e-03 -2.23469905e+01  5.91944025e+01  3.72331739e+01
  1.13048492e+01  1.13534511e-04  1.68163180e-06  3.11177128e-01
  2.77175593e-03 -2.16102416e-01  5.71734570e-05 -5.80056453e+01
 -9.23705556e-13  1.81934446e-01  1.74145518e-03 -5.73703156e-04
  1.18620635e-04 -6.33397531e-05 -1.22067054e+02 -9.56540965e+00]
```

In [28]:
```
#Predicting the test data

predictions = lm.predict(X_test)
```

In [29]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[29]:
```
Text(0, 0.5, 'Predicted Y')
```



In [30]:
```
#Evaluating the model

from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 61.1079728984134
MSE: 17358.992344245144
RMSE: 131.7535287734076
```

Decision Tree

In [31]:
```
#Create an instance of decision tree and fit it to the training data

dtree = DecisionTreeRegressor()
```

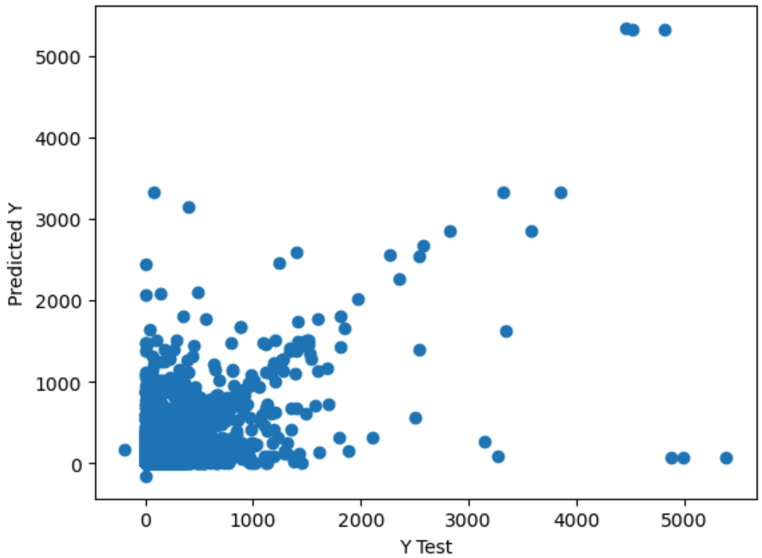
In [32]:
```
#fiitting the model
dtree.fit(X_train, y_train)
```

Out[32]:
```
▾ DecisionTreeRegressor i ?
DecisionTreeRegressor()
```
(https://scikit-learn.org/1.4/modules/generated/sklearn.tree.DecisionTreeRegressor.html)

In [33]:
```
#Predicting the test data

predictions = dtree.predict(X_test)
```

In [34]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[34]: Text(0, 0.5, 'Predicted Y')



In [35]:
```
#Evaluating the model

from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 45.578826633021734
MSE: 21751.73412857716
RMSE: 147.48469116683657
```

Gradient Boosting Regressor

In [36]:
```
from sklearn.ensemble import GradientBoostingRegressor
```

In [37]:
```
#Create an instance of gradient boosting model and fit it to the training data
gradient_boosting_model = GradientBoostingRegressor()
gradient_boosting_model.fit(X_train, y_train)
```

Out[37]:

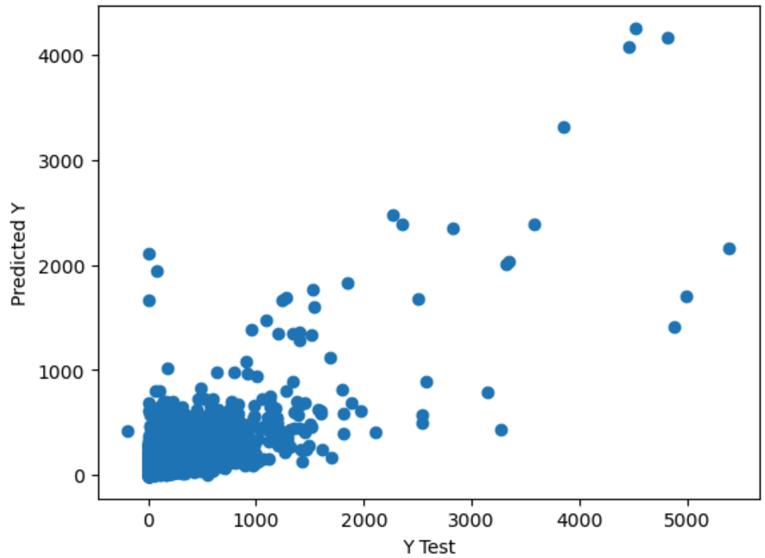
▾ GradientBoostingRegressor (https://scikit-learn.org/1.4/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html)

GradientBoostingRegressor()

In [38]:
```
#Predicting the test data

predictions = gradient_boosting_model.predict(X_test)
```

In [39]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[39]: Text(0, 0.5, 'Predicted Y')



In [40]:
```
#Evaluating the model
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 51.454967711157295
MSE: 14395.119153929398
RMSE: 119.9796614177978
```

XG Boost Model

In [41]:
```
#Create an instance of XGBoost and fit it to the training data
XGBoost = XGBRegressor()
```

In [42]:
```
XGBoost.fit(X_train, y_train)
```

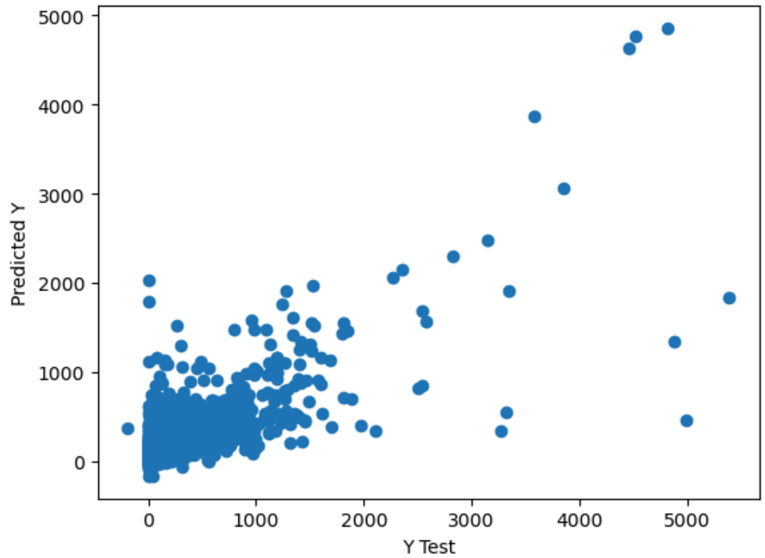
Out[42]:

```
XGBRegressor

XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=None, n_jobs=None,
```

In [43]:
```
#Predicting the test data

predictions = XGBoost.predict(X_test)
```

In [44]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[44]: Text(0, 0.5, 'Predicted Y')



In [45]:
```
#Evaluating the model
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 37.962528811721334
MSE: 12014.54931523065
RMSE: 109.61089961874526
```

LightGBM Model

In [46]:
```
#Create an instance of LightGBM Model and fit it to the training data
LightGBM_Model = LGBMRegressor()
```

In [47]:
```
LightGBM_Model.fit(X_train, y_train)
```

```
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.012769 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 2655
[LightGBM] [Info] Number of data points in the train set: 69748, number of used features: 18
[LightGBM] [Info] Start training from score 75.523513
```

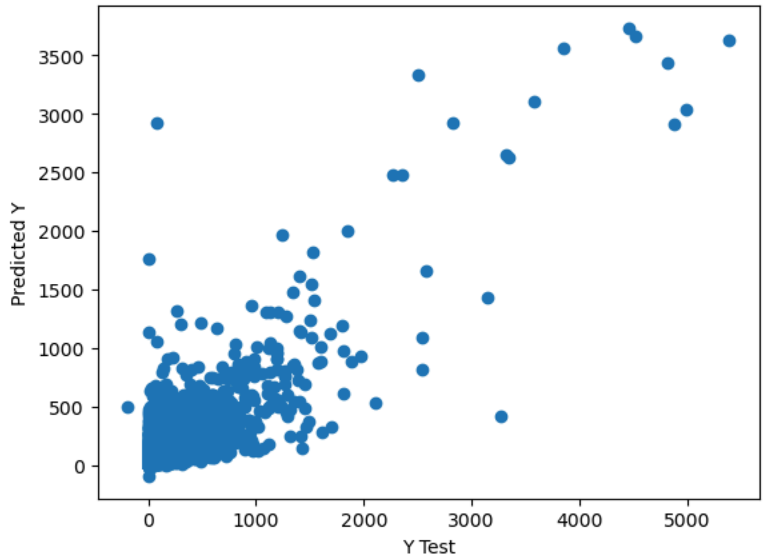
Out[47]:

```
▾ LGBMRegressor
LGBMRegressor()
```

In [48]:
```
#Predicting the test data

predictions = LightGBM_Model.predict(X_test)
```

In [49]:
```
plt.scatter(y_test,predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
```

Out[49]: Text(0, 0.5, 'Predicted Y')



In [50]:
```
#Evaluating the model
from sklearn import metrics

print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 43.65132654925742
MSE: 10525.256339592257
RMSE: 102.59267195853833
```