In [23]:

```python
import pandas as pd
```

In [24]:

```python
data = pd.read_csv('airline.csv')
```

In [25]:

```python
data.head()
```

Out[25]:

	Date	Passengers
0	01-01-1949	112
1	01-02-1949	118
2	01-03-1949	132
3	01-04-1949	129
4	01-05-1949	121

In [26]:

```python
data.tail()
```

Out[26]:

	Date	Passengers
139	10-08-1960	606
140	01-09-1960	508
141	01-10-1960	461
142	01-11-1960	390
143	01-12-1960	432

In [27]:

```python
data.shape
```

Out[27]:

```
(144, 2)
```

In [28]:

```
data.columns
```

Out[28]:

```
Index(['Date', 'Passengers'], dtype='object')
```

In [29]:

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 144 entries, 0 to 143
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   Date        144 non-null    object
 1   Passengers  144 non-null    int64 
dtypes: int64(1), object(1)
memory usage: 2.4+ KB
```

In [30]:

```
data.describe()
```

Out[30]:

	Passengers
count	144.000000
mean	280.298611
std	119.966317
min	104.000000
25%	180.000000
50%	265.500000
75%	360.500000
max	622.000000

In [31]:

```
data.isnull().sum()
```

Out[31]:

```
Date          0
Passengers    0
dtype: int64
```

In [32]:

```
data['Date'] = pd.to_datetime(data['Date'])
data.head()
```

Out[32]:

	Date	Passengers
0	1949-01-01	112
1	1949-01-02	118
2	1949-01-03	132
3	1949-01-04	129
4	1949-01-05	121

In [33]:

```
data_new = data.set_index('Date')
```

In [34]:

```
data_new.head()
```

Out[34]:

	Passengers
Date	
1949-01-01	112
1949-01-02	118
1949-01-03	132
1949-01-04	129
1949-01-05	121

In [35]:

```
split_date = '01-Dec-1957'
data_train = data_new.loc[data_new.index <= split_date].copy()
data_test = data_new.loc[data_new.index > split_date].copy()
```

In [36]:

```
data_train.shape
```

Out[36]:

```
(108, 1)
```

In [37]:

```
data_test.shape
```

Out[37]:

```
(36, 1)
```

In [38]:

```
def create_features(df, label=None):
    df['date'] = df.index
    df['hour'] = df['date'].dt.hour
    df['dayofweek'] = df['date'].dt.dayofweek
    df['quarter'] = df['date'].dt.quarter
    df['month'] = df['date'].dt.month
    df['year'] = df['date'].dt.year
    df['dayofyear'] = df['date'].dt.dayofyear
    df['dayofmonth'] = df['date'].dt.day
    df['weekofyear'] = df['date'].dt.weekofyear

    X = df[['hour','dayofweek','quarter','month','year',
           'dayofyear','dayofmonth','weekofyear']]
    if label:
        y = df[label]
        return X, y
    return X
```

In [39]:

```
X_train, y_train = create_features(data_train, label='Passengers')
X_test, y_test = create_features(data_test, label='Passengers')
X_train
```

```
<ipython-input-38-d2434dddca07>:10: FutureWarning: Series.dt.weekofyear an
d Series.dt.week have been deprecated. Please use Series.dt.isocalendar().
week instead.
  df['weekofyear'] = df['date'].dt.weekofyear
<ipython-input-38-d2434dddca07>:10: FutureWarning: Series.dt.weekofyear an
d Series.dt.week have been deprecated. Please use Series.dt.isocalendar().
week instead.
  df['weekofyear'] = df['date'].dt.weekofyear
```

Out[39]:

	hour	dayofweek	quarter	month	year	dayofyear	dayofmonth	weekofyear
Date								
1949-01-01	0	5	1	1	1949	1	1	53
1949-01-02	0	6	1	1	1949	2	2	53
1949-01-03	0	0	1	1	1949	3	3	1
1949-01-04	0	1	1	1	1949	4	4	1
1949-01-05	0	2	1	1	1949	5	5	1
...	...	...	...	...	...	...	...	...
1957-01-08	0	1	1	1	1957	8	8	2
1957-01-09	0	2	1	1	1957	9	9	2
1957-01-10	0	3	1	1	1957	10	10	2
1957-01-11	0	4	1	1	1957	11	11	2
1957-01-12	0	5	1	1	1957	12	12	2

108 rows × 8 columns

In [40]:

```
import xgboost as xgb
from xgboost import plot_importance, plot_tree
from sklearn.metrics import mean_squared_error, mean_absolute_error
```

In [41]:

```
reg = xgb.XGBRegressor(n_estimators=1000)
```

In [42]:

```
reg.fit(X_train, y_train,
        eval_set=[(X_train, y_train), (X_test, y_test)],
        early_stopping_rounds=50,
       verbose=False)
```

```
c:\python\lib\site-packages\xgboost\sklearn.py:793: UserWarning: `early_st
opping_rounds` in `fit` method is deprecated for better compatibility with
scikit-learn, use `early_stopping_rounds` in constructor or`set_params` in
stead.
  warnings.warn(
```

Out[42]:

```
XGBRegressor(base_score=0.5, booster='gbtree', callbacks=None,
             colsample_bylevel=1, colsample_bynode=1, colsample_bytree=1,
             early_stopping_rounds=None, enable_categorical=False,
             eval_metric=None, gamma=0, gpu_id=-1, grow_policy='depthwis
e',
             importance_type=None, interaction_constraints='',
             learning_rate=0.300000012, max_bin=256, max_cat_to_onehot=4,
             max_delta_step=0, max_depth=6, max_leaves=0, min_child_weight
=1,
             missing=nan, monotone_constraints='()', n_estimators=1000,
             n_jobs=0, num_parallel_tree=1, predictor='auto', random_state
=0,
             reg_alpha=0, reg_lambda=1, ...)
```

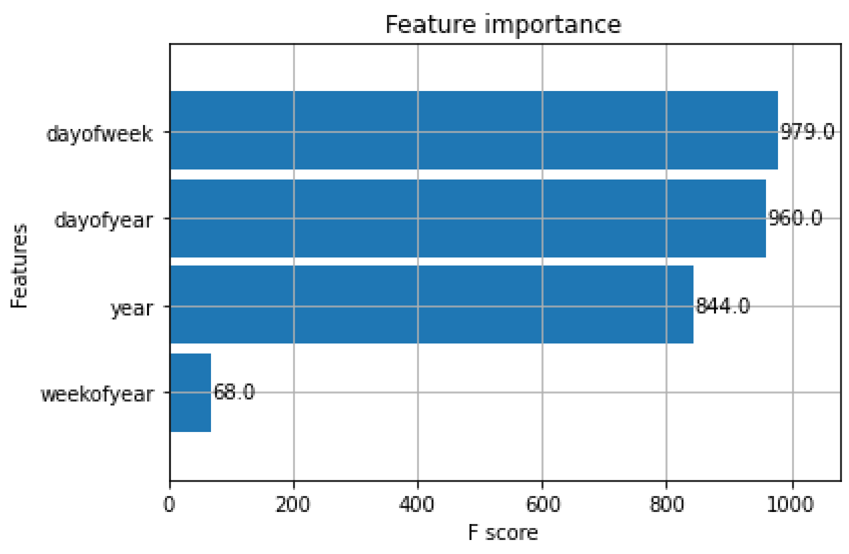
In [44]:

```
print("Training Accuracy :", reg.score(X_train, y_train))
```

```
Training Accuracy : 0.9999624598552331
```

In [45]:

```
_ = plot_importance(reg, height=0.9)
```

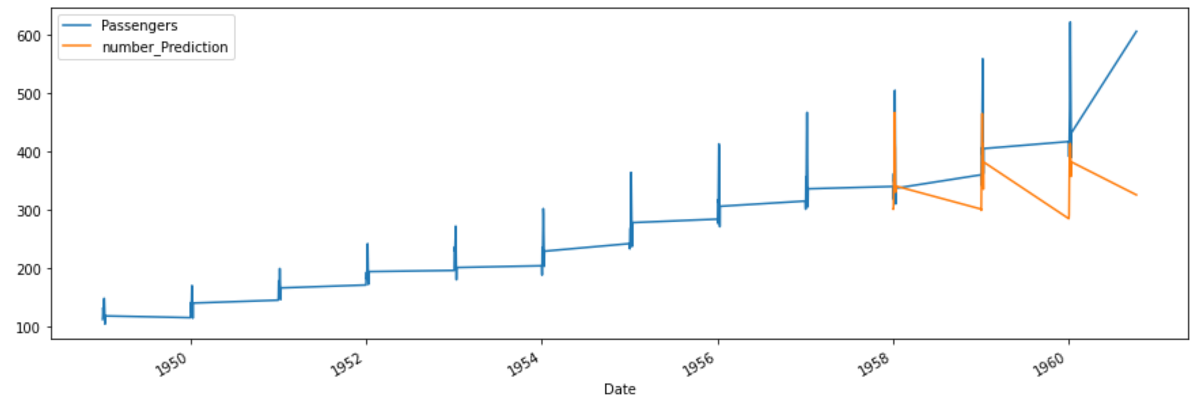


In [46]:

```
data_test['number_Prediction'] = reg.predict(X_test)
data_all = pd.concat([data_test, data_train], sort=False)
```

In [47]:

```
_ = data_all[['Passengers','number_Prediction']].plot(figsize=(15, 5))
```


Passengers
number_Prediction
Date