Project Objective

Build a simple regression model to predict the annual salary for each customer using the attributes you identified above

How accurate is your model? Should ANZ use it to segment customers (for whom it does not have this data) into income brackets for reporting purposes?

For a challenge: build a decision-tree based model to predict salary. Does it perform better? How would you accurately test the performance of this model?

Import Libraries

In [1]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn
import shap

import xgboost as xgb
from xgboost import XGBClassifier, XGBRegressor
from xgboost import to_graphviz, plot_importance

%matplotlib inline
sns.set_style('dark')
sns.set(font_scale=1.2)

from sklearn.linear_model import LinearRegression

from sklearn.inspection import permutation_importance
from sklearn.model_selection import cross_val_score, train_test_split, GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler, MinMaxScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.metrics import confusion_matrix, classification_report, mean_absolute_error, mean_squared_
error,r2_score
from sklearn.metrics import plot_confusion_matrix, plot_precision_recall_curve, plot_roc_curve, accurac
y_score
from sklearn.metrics import auc, f1_score, precision_score, recall_score, roc_auc_score

import warnings
warnings.filterwarnings('ignore')

import pickle
from pickle import dump, load

np.random.seed(0)

#from pycaret.classification import *
#from pycaret.clustering import *
from pycaret.regression import *

pd.set_option('display.max_columns',100)
#pd.set_option('display.max_rows',100)
pd.set_option('display.width', 1000)
np.set_printoptions(suppress=True)
```

Data Exploration and Analysis

In [2]:
```python
df = pd.read_csv("salary3.csv")
```

In [3]:
```python
df
```

Out[3]:

	account	first_name	age	gender	balance	months	weekspayment	annual
0	ACC-1037050564	Rhonda	40	F	51472.20	12	52	46388.68
1	ACC-1056639002	Michael	22	M	298308.49	12	24	166140.52
2	ACC-1199531521	Billy	52	M	86898.05	12	28	196860.56
3	ACC-1217063613	Kimberly	27	F	13769.63	8	8	252908.24
4	ACC-1222300524	Michael	38	M	22826.60	12	52	52110.76
...	...	...	...	...	...	...	...	...
95	ACC-854938045	James	28	M	88169.32	12	28	132011.36
96	ACC-90814749	Christopher	35	M	65301.33	12	24	120050.84
97	ACC-958000567	Sandra	34	F	59807.25	12	28	182915.72
98	ACC-964839203	Michael	21	M	467645.22	12	52	81130.40
99	ACC-966140392	Joseph	21	M	49877.75	12	28	133791.32

100 rows × 8 columns

In [4]:
```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 8 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   account       100 non-null    object 
 1   first_name    100 non-null    object 
 2   age           100 non-null    int64  
 3   gender        100 non-null    object 
 4   balance       100 non-null    float64
 5   months        100 non-null    int64  
 6   weekspayment  100 non-null    int64  
 7   annual        100 non-null    float64
dtypes: float64(2), int64(3), object(3)
memory usage: 6.4+ KB
```

In [5]:
```python
df.describe(include='all')
```

Out[5]:

	account	first_name	age	gender	balance	months	weekspayment	annual
count	100	100	100.000000	100	1.000000e+02	100.000000	100.000000	100.000000
unique	100	80	NaN	2	NaN	NaN	NaN	NaN
top	ACC-3541460373	Michael	NaN	M	NaN	NaN	NaN	NaN
freq	1	6	NaN	56	NaN	NaN	NaN	NaN
mean	NaN	NaN	31.770000	NaN	1.434244e+05	11.840000	35.320000	119458.242800
std	NaN	NaN	11.544254	NaN	2.350789e+05	0.787786	14.606543	72684.766459
min	NaN	NaN	18.000000	NaN	1.376963e+04	8.000000	8.000000	29952.000000
25%	NaN	NaN	22.000000	NaN	4.965724e+04	12.000000	24.000000	59972.120000
50%	NaN	NaN	29.500000	NaN	7.227742e+04	12.000000	28.000000	101370.360000
75%	NaN	NaN	39.250000	NaN	1.150409e+05	12.000000	52.000000	150109.700000
max	NaN	NaN	78.000000	NaN	1.584768e+06	12.000000	56.000000	459470.960000

In [6]:
```python
df.shape
```

Out[6]: (100, 8)

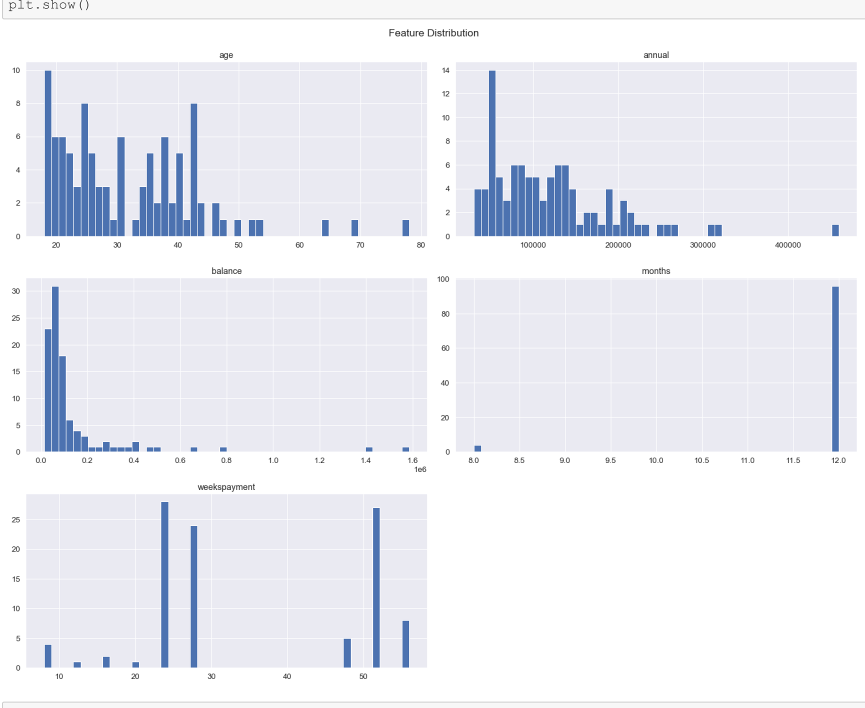
In [7]:
```python
df.columns
```

Out[7]:
```
Index(['account', 'first_name', 'age', 'gender', 'balance', 'months', 'weekspayment', 'annual'], dtyp
e='object')
```

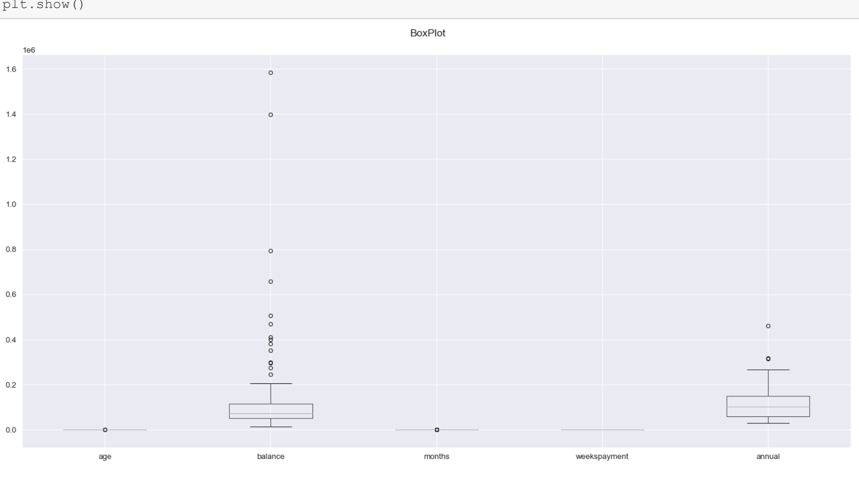
Data Visualization

Univariate Data Exploration

In [8]:
```python
df.hist(bins=50, figsize=(20,15))
plt.suptitle('Feature Distribution', x=0.5, y=1.02, ha='center', fontsize='large')
plt.tight_layout()
plt.show()
```



In [9]:
```python
df.boxplot(figsize=(20,10))
plt.suptitle('BoxPlot', x=0.5, y=1.02, ha='center', fontsize='large')
plt.tight_layout()
plt.show()
```



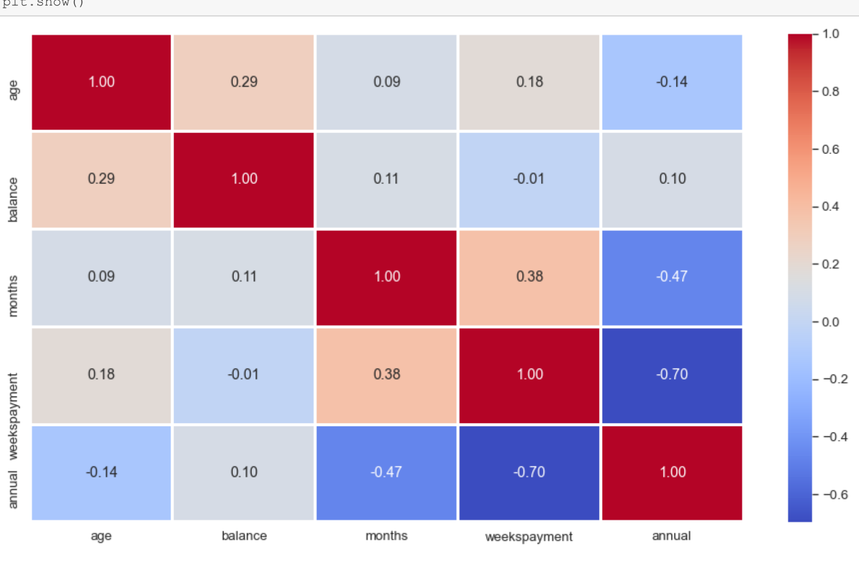
Correlation

In [10]:
```python
df.corr()
```

Out[10]:

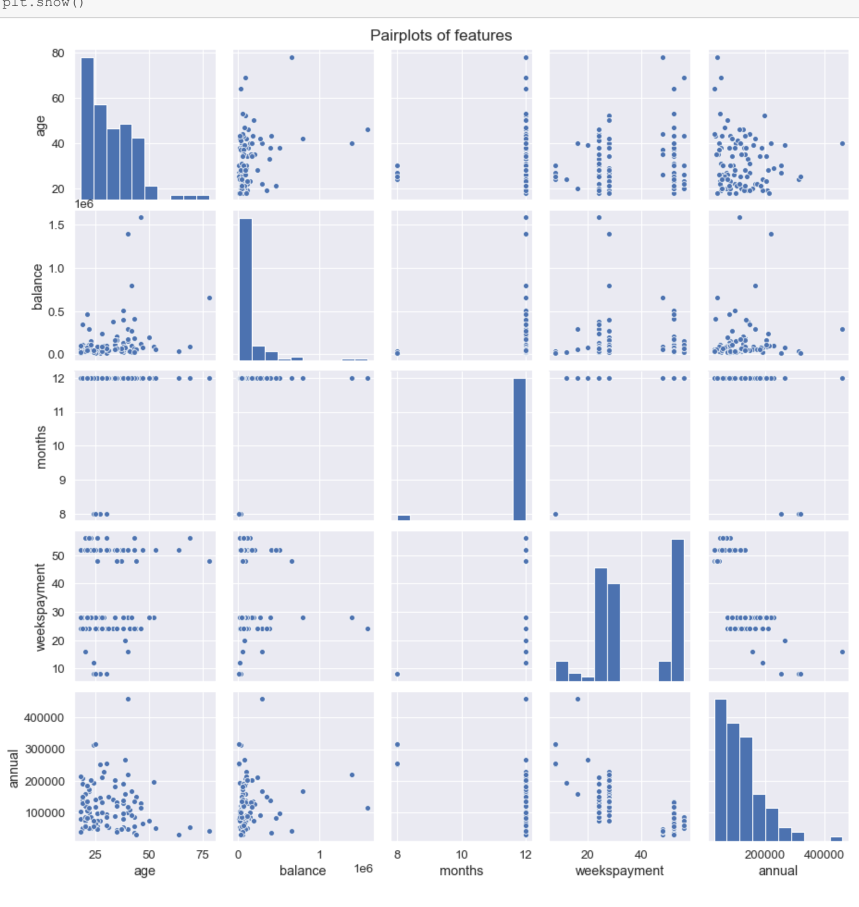
	age	balance	months	weekspayment	annual
age	1.000000	0.289224	0.093653	0.182368	-0.135264
balance	0.289224	1.000000	0.107047	-0.013461	0.101847
months	0.093653	0.107047	1.000000	0.383716	-0.466623
weekspayment	0.182368	-0.013461	0.383716	1.000000	-0.696428
annual	-0.135264	0.101847	-0.466623	-0.696428	1.000000

In [11]:
```python
plt.figure(figsize=(16,9))
sns.heatmap(df.corr(),cmap="coolwarm",annot=True,fmt='.2f',linewidths=2)
plt.show()
```



Pairplots

In [12]:
```python
sns.pairplot(df)
plt.suptitle('Pairplots of features', x=0.5, y=1.02, ha='center', fontsize='large')
plt.show()
```



Data Preprocessing

Treat Missing Values

In [13]:
```python
df.isnull().sum()
```

Out[13]:
```
account         0
first_name      0
age             0
gender          0
balance         0
months          0
weekspayment    0
annual          0
dtype: int64
```

Treat Duplicate Values

In [14]:
```python
df.duplicated(keep='first').sum()
```

Out[14]: 0

Treat Outliers

In [15]:
```python
df.describe()
```

Out[15]:

	age	balance	months	weekspayment	annual
count	100.000000	1.000000e+02	100.000000	100.000000	100.000000
mean	31.770000	1.434244e+05	11.840000	35.320000	119458.242800
std	11.544254	2.350789e+05	0.787786	14.606543	72684.766459
min	18.000000	1.376963e+04	8.000000	8.000000	29952.000000
25%	22.000000	4.965724e+04	12.000000	24.000000	59972.120000
50%	29.500000	7.227742e+04	12.000000	28.000000	101370.360000
75%	39.250000	1.150409e+05	12.000000	52.000000	150109.700000
max	78.000000	1.584768e+06	12.000000	56.000000	459470.960000

In [16]:
```python
df.columns
```

Out[16]:
```
Index(['account', 'first_name', 'age', 'gender', 'balance', 'months', 'weekspayment', 'annual'], dtyp
e='object')
```

In [17]:
```python
df.drop(['account', 'first_name'],axis=1,inplace=True)
```

In [18]:
```python
df
```

Out[18]:

	age	gender	balance	months	weekspayment	annual
0	40	F	51472.20	12	52	46388.68
1	22	M	298308.49	12	24	166140.52
2	52	M	86898.05	12	28	196860.56
3	27	F	13769.63	8	8	252908.24
4	38	M	22826.60	12	52	52110.76
...	...	...	...	...	...	...
95	28	M	88169.32	12	28	132011.36
96	35	M	65301.33	12	24	120050.84
97	34	F	59807.25	12	28	182915.72
98	21	M	467645.22	12	52	81130.40
99	21	M	49877.75	12	28	133791.32

100 rows × 6 columns

Perform One-Hot Encoding

In [19]:
```python
df2 = pd.get_dummies(df, drop_first=True)
```

In [20]:
```python
df2
```

Out[20]:

	age	balance	months	weekspayment	annual	gender_M
0	40	51472.20	12	52	46388.68	0
1	22	298308.49	12	24	166140.52	1
2	52	86898.05	12	28	196860.56	1
3	27	13769.63	8	8	252908.24	0
4	38	22826.60	12	52	52110.76	1
...	...	...	...	...	...	...
95	28	88169.32	12	28	132011.36	1
96	35	65301.33	12	24	120050.84	1
97	34	59807.25	12	28	182915.72	0
98	21	467645.22	12	52	81130.40	1
99	21	49877.75	12	28	133791.32	1

100 rows × 6 columns

In [21]:
```python
df2.columns
```

Out[21]:
```
Index(['age', 'balance', 'months', 'weekspayment', 'annual', 'gender_M'], dtype='object')
```

In [22]:
```python
df2 = df2[['age', 'balance', 'months', 'weekspayment','gender_M','annual']]
```

In [23]:
```python
df2
```

Out[23]:

	age	balance	months	weekspayment	gender_M	annual
0	40	51472.20	12	52	0	46388.68
1	22	298308.49	12	24	1	166140.52
2	52	86898.05	12	28	1	196860.56
3	27	13769.63	8	8	0	252908.24
4	38	22826.60	12	52	1	52110.76
...	...	...	...	...	...	...
95	28	88169.32	12	28	1	132011.36
96	35	65301.33	12	24	1	120050.84
97	34	59807.25	12	28	0	182915.72
98	21	467645.22	12	52	1	81130.40
99	21	49877.75	12	28	1	133791.32

100 rows × 6 columns

Treat Data Types

In [24]:
```python
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   age           100 non-null    int64  
 1   balance       100 non-null    float64
 2   months        100 non-null    int64  
 3   weekspayment  100 non-null    int64  
 4   gender_M      100 non-null    uint8  
 5   annual        100 non-null    float64
dtypes: float64(2), int64(3), uint8(1)
memory usage: 4.1 KB
```

Create and save processed dataset

In [25]:
```python
#df2.to_csv("train.csv",index=False)
```

Train Test Split

In [26]:
```python
df2.shape
```

Out[26]: (100, 6)

In [27]:
```python
X = df2.iloc[:,0:5]
y = df2.iloc[:,5]
```

In [28]:
```python
X.values, y.values
```

Out[28]:
```
(array([[     40.  ,   51472.2 ,      12.  ,      52.  ,       0.  ],
        [     22.  ,  298308.49,      12.  ,      24.  ,       1.  ],
        [     52.  ,   86898.05,      12.  ,      28.  ,       1.  ],
        [     27.  ,   13769.63,       8.  ,       8.  ,       0.  ],
        [     38.  ,   22826.6 ,      12.  ,      52.  ,       1.  ],
        [     42.  ,   28575.85,      12.  ,      24.  ,       1.  ],
        [     44.  ,   49934.57,      12.  ,      24.  ,       1.  ],
        [     33.  ,  381472.24,      12.  ,      24.  ,       1.  ],
        [     46.  ,  122866.07,      12.  ,      24.  ,       0.  ],
        [     39.  ,   89270.66,      12.  ,      28.  ,       1.  ],
        [     22.  ,   37087.99,      12.  ,      24.  ,       1.  ],
        [     35.  ,   57016.46,      12.  ,      52.  ,       0.  ],
        [     31.  ,   51803.62,      12.  ,      52.  ,       1.  ],
        [     43.  ,  183287.54,      12.  ,      52.  ,       1.  ],
        [     21.  ,   31881.05,      12.  ,      28.  ,       0.  ],
        [     40.  ,  295834.6 ,      12.  ,      16.  ,       1.  ],
        [     18.  ,   17476.44,      12.  ,      24.  ,       0.  ],
        [     26.  ,   29963.79,      12.  ,      56.  ,       0.  ],
        [     20.  ,   40760.94,      12.  ,      24.  ,       1.  ],
        [     23.  ,  150375.3 ,      12.  ,      56.  ,       0.  ],
        [     38.  ,  128339.74,      12.  ,      24.  ,       0.  ],
        [     38.  ,   43956.52,      12.  ,      24.  ,       1.  ],
        [     19.  ,   88893.08,      12.  ,      28.  ,       0.  ],
        [     21.  ,   48462.09,      12.  ,      28.  ,       0.  ],
        [     20.  ,   97942.75,      12.  ,      56.  ,       0.  ],
        [     27.  ,   50023.1 ,      12.  ,      28.  ,       0.  ],
        [     38.  ,   92901.74,      12.  ,      24.  ,       1.  ],
        [     30.  ,   25328.81,      12.  ,      24.  ,       1.  ],
        [     26.  ,   78678.67,      12.  ,      56.  ,       1.  ],
        [     21.  ,   72529.42,      12.  ,      52.  ,       0.  ],
        [     42.  ,  105414.15,      12.  ,      24.  ,       1.  ],
        [     46.  , 1584768.28,      12.  ,      24.  ,       1.  ],
        [     22.  ,   48995.72,      12.  ,      24.  ,       1.  ],
        [     30.  ,   14129.58,       8.  ,       8.  ,       0.  ],
        [     24.  ,   20776.76,      12.  ,      12.  ,       0.  ],
        [     19.  ,   93731.43,      12.  ,      28.  ,       1.  ],
        [     27.  ,   95447.77,      12.  ,      52.  ,       1.  ],
        [     35.  ,  204657.85,      12.  ,      52.  ,       0.  ],
        [     19.  ,  114059.67,      12.  ,      52.  ,       1.  ],
        [     19.  ,   75015.58,      12.  ,      24.  ,       0.  ],
        [     26.  ,   27918.85,      12.  ,      24.  ,       0.  ],
        [     20.  ,   53532.66,      12.  ,      16.  ,       1.  ],
        [     18.  ,   55894.36,      12.  ,      52.  ,       0.  ],
        [     29.  ,  106299.1 ,      12.  ,      28.  ,       1.  ],
        [     40.  ,  151094.12,      12.  ,      52.  ,       0.  ],
        [     43.  ,  409793.72,      12.  ,      52.  ,       0.  ],
        [     42.  ,  792776.29,      12.  ,      28.  ,       1.  ],
        [     24.  ,   38285.3 ,       8.  ,       8.  ,       1.  ],
        [     50.  ,  198251.88,      12.  ,      28.  ,       0.  ],
        [     31.  ,   50652.61,      12.  ,      24.  ,       1.  ],
        [     43.  ,   72949.87,      12.  ,      56.  ,       0.  ],
        [     53.  ,   61342.65,      12.  ,      52.  ,       0.  ],
        [     37.  ,   39874.67,      12.  ,      52.  ,       1.  ],
        [     23.  ,   66750.5 ,      12.  ,      52.  ,       1.  ],
        [     22.  ,   59122.64,      12.  ,      28.  ,       1.  ],
        [     69.  ,   94283.29,      12.  ,      56.  ,       1.  ],
        [     26.  ,   58334.28,      12.  ,      24.  ,       1.  ],
        [     44.  ,   63183.45,      12.  ,      48.  ,       1.  ],
        [     30.  ,  105638.82,      12.  ,      52.  ,       0.  ],
        [     19.  ,   57364.72,      12.  ,      28.  ,       0.  ],
        [     25.  ,   16864.59,       8.  ,       8.  ,       1.  ],
        [     34.  ,  114575.08,      12.  ,      52.  ,       0.  ],
        [     35.  ,  164255.24,      12.  ,      52.  ,       1.  ],
        [     22.  ,  116438.43,      12.  ,      56.  ,       1.  ],
        [     25.  ,   67715.64,      12.  ,      52.  ,       0.  ],
        [     34.  ,  166920.02,      12.  ,      28.  ,       0.  ],
        [     41.  ,   32786.24,      12.  ,      24.  ,       0.  ],
        [     39.  ,   78807.94,      12.  ,      20.  ,       0.  ],
        [     24.  ,   27141.67,      12.  ,      28.  ,       1.  ],
        [     20.  ,   76688.05,      12.  ,      52.  ,       0.  ],
        [     25.  ,   64581.35,      12.  ,      28.  ,       0.  ],
        [     18.  ,   98107.03,      12.  ,      28.  ,       1.  ],
        [     20.  ,   58654.59,      12.  ,      52.  ,       1.  ],
        [     28.  ,  244401.96,      12.  ,      24.  ,       1.  ],
        [     40.  , 1398902.55,      12.  ,      28.  ,       0.  ],
        [     37.  ,   72025.43,      12.  ,      48.  ,       1.  ],
        [     38.  ,  506145.72,      12.  ,      52.  ,       1.  ],
        [     25.  ,   54667.71,      12.  ,      24.  ,       1.  ],
        [     47.  ,   75920.32,      12.  ,      52.  ,       0.  ],
        [     38.  ,  399484.97,      12.  ,      28.  ,       1.  ],
        [     21.  ,   83700.42,      12.  ,      28.  ,       1.  ],
        [     43.  ,   26484.2 ,      12.  ,      24.  ,       0.  ],
        [     26.  ,   94134.87,      12.  ,      48.  ,       0.  ],
        [     24.  ,   51624.95,      12.  ,      52.  ,       0.  ],
        [     20.  ,   42000.3 ,      12.  ,      24.  ,       1.  ],
        [     42.  ,  275038.66,      12.  ,      28.  ,       1.  ],
        [     35.  ,   54704.11,      12.  ,      48.  ,       0.  ],
        [     64.  ,   36491.38,      12.  ,      52.  ,       0.  ],
        [     18.  ,   26536.67,      12.  ,      24.  ,       0.  ],
        [     23.  ,  111215.39,      12.  ,      28.  ,       1.  ],
        [     40.  ,  171879.76,      12.  ,      52.  ,       1.  ],
        [     78.  ,  657477.5 ,      12.  ,      48.  ,       1.  ],
        [     28.  ,   35050.32,      12.  ,      24.  ,       1.  ],
        [     30.  ,   71436.2 ,      12.  ,      56.  ,       0.  ],
        [     19.  ,  350505.83,      12.  ,      24.  ,       1.  ],
        [     28.  ,   88169.32,      12.  ,      28.  ,       1.  ],
        [     35.  ,   65301.33,      12.  ,      24.  ,       1.  ],
        [     34.  ,   59807.25,      12.  ,      28.  ,       0.  ],
        [     21.  ,  467645.22,      12.  ,      52.  ,       1.  ],
        [     21.  ,   49877.75,      12.  ,      28.  ,       1.  ]]),
 array([ 46388.68, 166140.52, 196860.56, 252908.24,  52110.76,  87442.16,
        150141.68, 141362.52, 128463.4 , 159699.28, 100306.44,  44873.4 ,
         51804.48, 134576.52, 119944.76, 459470.96,  79959.36,  52710.84,
        120655.6 ,  84576.96,  80138.24, 130000.  , 206827.92, 105231.36,
         72565.48, 147687.8 , 189774.52,  94048.24,  57143.32,  53927.64,
         91406.12, 113357.92, 115373.96, 255366.8 , 193053.12, 137267.52,
         72603.96, 127048.48,  51508.6 , 191200.88,  95843.28, 157401.4 ,
         39589.16, 229075.6 , 101221.64,  34550.36, 168025.52, 313273.48,
         74722.96, 150099.04,  55538.08,  50464.44,  47707.4 ,  51100.92,
        173096.04,  54242.24,  74566.96,  31009.16,  85109.44, 129239.24,
        317575.96,  88992.28, 132327.52,  51134.72,  63717.16, 202993.96,
        109278.52, 265382.52,  73220.16,  59217.08, 118682.72, 214875.96,
         57184.4 , 210848.04, 219234.08,  47671.  ,  97809.4 , 139768.2 ,
         66168.44, 137663.76, 186089.8 ,  84778.72,  47876.92,  55111.68,
         85323.16,  91406.64,  37716.64,  29952.  , 101519.08, 203005.4 ,
        118578.72,  40685.84,  99658.52,  60223.8 , 148446.48, 132011.36,
        120050.84, 182915.72,  81130.4 , 133791.32]))
```

In [29]:
```python
X_train, X_test, y_train, y_test = train_test_split(X.values, y.values, test_size=0.2, random_state=0)
```

In [30]:
```python
X_train.shape, X_test.shape, y_train.shape, y_test.shape
```

Out[30]: ((80, 5), (20, 5), (80,), (20,))

Feature Scaling

In [31]:
```python
X_train
```

Out[31]:
```
array([[     29.  ,  106299.1 ,      12.  ,      28.  ,       1.  ],
       [     35.  ,  164255.24,      12.  ,      52.  ,       1.  ],
       [     27.  ,   13769.63,       8.  ,       8.  ,       0.  ],
       [     18.  ,   98107.03,      12.  ,      28.  ,       1.  ],
       [     43.  ,  409793.72,      12.  ,      52.  ,       0.  ],
       [     50.  ,  198251.88,      12.  ,      28.  ,       0.  ],
       [     44.  ,   49934.57,      12.  ,      24.  ,       1.  ],
       [     21.  ,   49877.75,      12.  ,      28.  ,       1.  ],
       [     26.  ,   94134.87,      12.  ,      48.  ,       0.  ],
       [     38.  ,  506145.72,      12.  ,      52.  ,       1.  ],
       [     25.  ,   16864.59,       8.  ,       8.  ,       1.  ],
       [     21.  ,   83700.42,      12.  ,      28.  ,       1.  ],
       [     40.  ,  171879.76,      12.  ,      52.  ,       1.  ],
       [     24.  ,   27141.67,      12.  ,      28.  ,       1.  ],
       [     53.  ,   61342.65,      12.  ,      52.  ,       0.  ],
       [     30.  ,   25328.81,      12.  ,      24.  ,       1.  ],
       [     20.  ,   40760.94,      12.  ,      24.  ,       1.  ],
       [     26.  ,   58334.28,      12.  ,      24.  ,       1.  ],
       [     22.  ,  116438.43,      12.  ,      56.  ,       1.  ],
       [     40.  , 1398902.55,      12.  ,      28.  ,       0.  ],
       [     22.  ,  298308.49,      12.  ,      24.  ,       1.  ],
       [     34.  ,  114575.08,      12.  ,      52.  ,       0.  ],
       [     18.  ,   55894.36,      12.  ,      52.  ,       0.  ],
       [     20.  ,   53532.66,      12.  ,      16.  ,       1.  ],
       [     38.  ,   22826.6 ,      12.  ,      52.  ,       1.  ],
       [     40.  ,  295834.6 ,      12.  ,      16.  ,       1.  ],
       [     26.  ,   29963.79,      12.  ,      56.  ,       0.  ],
       [     26.  ,   27918.85,      12.  ,      24.  ,       0.  ],
       [     19.  ,  114059.67,      12.  ,      52.  ,       1.  ],
       [     42.  ,   28575.85,      12.  ,      24.  ,       1.  ],
       [     78.  ,  657477.5 ,      12.  ,      48.  ,       1.  ],
       [     19.  ,   57364.72,      12.  ,      28.  ,       0.  ],
       [     40.  ,   51472.2 ,      12.  ,      52.  ,       0.  ],
       [     24.  ,   20776.76,      12.  ,      12.  ,       0.  ],
       [     26.  ,   78678.67,      12.  ,      56.  ,       1.  ],
       [     43.  ,   72949.87,      12.  ,      56.  ,       0.  ],
       [     35.  ,   57016.46,      12.  ,      52.  ,       0.  ],
       [     19.  ,   93731.43,      12.  ,      28.  ,       1.  ],
       [     21.  ,   48462.09,      12.  ,      28.  ,       0.  ],
       [     37.  ,   39874.67,      12.  ,      52.  ,       1.  ],
       [     22.  ,   37087.99,      12.  ,      24.  ,       1.  ],
       [     46.  , 1584768.28,      12.  ,      24.  ,       1.  ],
       [     41.  ,   32786.24,      12.  ,      24.  ,       0.  ],
       [     44.  ,   63183.45,      12.  ,      48.  ,       1.  ],
       [     38.  ,  399484.97,      12.  ,      28.  ,       1.  ],
       [     42.  ,  275038.66,      12.  ,      28.  ,       1.  ],
       [     22.  ,   48995.72,      12.  ,      24.  ,       1.  ],
       [     20.  ,   42000.3 ,      12.  ,      24.  ,       1.  ],
       [     21.  ,   31881.05,      12.  ,      28.  ,       0.  ],
       [     23.  ,  111215.39,      12.  ,      28.  ,       1.  ],
       [     23.  ,  150375.3 ,      12.  ,      56.  ,       0.  ],
       [     21.  ,   72529.42,      12.  ,      52.  ,       0.  ],
       [     31.  ,   50652.61,      12.  ,      24.  ,       1.  ],
       [     34.  ,   59807.25,      12.  ,      28.  ,       0.  ],
       [     21.  ,  467645.22,      12.  ,      52.  ,       1.  ],
       [     20.  ,   76688.05,      12.  ,      52.  ,       0.  ],
       [     38.  ,  128339.74,      12.  ,      24.  ,       0.  ],
       [     19.  ,  350505.83,      12.  ,      24.  ,       1.  ],
       [     20.  ,   58654.59,      12.  ,      52.  ,       1.  ],
       [     25.  ,   54667.71,      12.  ,      24.  ,       1.  ],
       [     27.  ,   50023.1 ,      12.  ,      28.  ,       0.  ],
       [     35.  ,  204657.85,      12.  ,      52.  ,       0.  ],
       [     43.  ,   26484.2 ,      12.  ,      24.  ,       0.  ],
       [     42.  ,  792776.29,      12.  ,      28.  ,       1.  ],
       [     19.  ,   75015.58,      12.  ,      24.  ,       0.  ],
       [     34.  ,  166920.02,      12.  ,      28.  ,       0.  ],
       [     30.  ,  105638.82,      12.  ,      52.  ,       0.  ],
       [     31.  ,   51803.62,      12.  ,      52.  ,       1.  ],
       [     18.  ,   26536.67,      12.  ,      24.  ,       0.  ],
       [     25.  ,   64581.35,      12.  ,      28.  ,       0.  ],
       [     64.  ,   36491.38,      12.  ,      52.  ,       0.  ],
       [     27.  ,   95447.77,      12.  ,      52.  ,       1.  ],
       [     38.  ,   43956.52,      12.  ,      24.  ,       1.  ],
       [     24.  ,   51624.95,      12.  ,      52.  ,       0.  ],
       [     39.  ,   89270.66,      12.  ,      28.  ,       1.  ],
       [     35.  ,   65301.33,      12.  ,      24.  ,       1.  ],
       [     39.  ,   78807.94,      12.  ,      20.  ,       0.  ],
       [     25.  ,   67715.64,      12.  ,      52.  ,       0.  ],
       [     24.  ,   38285.3 ,       8.  ,       8.  ,       1.  ],
       [     40.  ,  151094.12,      12.  ,      52.  ,       0.  ]])
```

In [32]:
```python
scaler = StandardScaler()
```

In [33]:
```python
X_train_scaled = scaler.fit_transform(X_train)
```

In [34]:
```python
X_test_scaled = scaler.transform(X_test)
```

In [35]:
```python
X_train_scaled
```

Out[35]:
```
array([[-0.18982307, -0.18390253,  0.19738551, -0.50337646,  0.90453403],
       [ 0.34931937,  0.0413495 ,  0.19738551,  1.15155984,  0.90453403],
       [-0.36953721, -0.5435271 , -5.06622805, -1.88249003, -1.1055416 ],
       [-1.17825087, -0.21574179,  0.19738551, -0.50337646,  0.90453403],
       [ 1.06817596,  0.9956581 ,  0.19738551,  1.15155984, -1.1055416 ],
       [ 1.69717547,  0.17348067,  0.19738551, -0.50337646, -1.1055416 ],
       [ 1.15803303, -0.40296863,  0.19738551, -0.77919917,  0.90453403],
       [-0.90867965, -0.40318946,  0.19738551, -0.50337646,  0.90453403],
       [-0.45939429, -0.23117997,  0.19738551,  0.87573712, -1.1055416 ],
       [ 0.61889059,  1.37013929,  0.19738551,  1.15155984,  0.90453403],
       [-0.54925136, -0.53149824, -5.06622805, -1.88249003,  0.90453403],
       [-0.90867965, -0.27173445,  0.19738551, -0.50337646,  0.90453403],
       [ 0.79860474,  0.07098292,  0.19738551,  1.15155984,  0.90453403],
       [-0.63910843, -0.4915554 ,  0.19738551, -0.50337646,  0.90453403],
       [ 1.96674669, -0.35863004,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.09996599, -0.49860125,  0.19738551, -0.77919917,  0.90453403],
       [-0.99853673, -0.43862281,  0.19738551, -0.77919917,  0.90453403],
       [-0.45939429, -0.37032236,  0.19738551, -0.77919917,  0.90453403],
       [-0.81882258, -0.14449507,  0.19738551,  1.42738255,  0.90453403],
       [ 0.79860474,  4.83992341,  0.19738551, -0.50337646, -1.1055416 ],
       [-0.81882258,  0.56236018,  0.19738551, -0.77919917,  0.90453403],
       [ 0.2594623 , -0.15173715,  0.19738551,  1.15155984, -1.1055416 ],
       [-1.17825087, -0.37980534,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.99853673, -0.38898431,  0.19738551, -1.3308446 ,  0.90453403],
       [ 0.61889059, -0.50832633,  0.19738551,  1.15155984,  0.90453403],
       [ 0.79860474,  0.55274517,  0.19738551, -1.3308446 ,  0.90453403],
       [-0.45939429, -0.48058696,  0.19738551,  1.42738255, -1.1055416 ],
       [-0.45939429, -0.48853481,  0.19738551, -0.77919917, -1.1055416 ],
       [-1.0883938 , -0.15374034,  0.19738551,  1.15155984,  0.90453403],
       [ 0.97831888, -0.48598132,  0.19738551, -0.77919917,  0.90453403],
       [ 4.21317352,  1.95830461,  0.19738551,  0.87573712,  0.90453403],
       [-1.0883938 , -0.37409065,  0.19738551, -0.50337646, -1.1055416 ],
       [ 0.79860474, -0.39699248,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.63910843, -0.51629322,  0.19738551, -1.60666732, -1.1055416 ],
       [-0.45939429, -0.29125196,  0.19738551,  1.42738255,  0.90453403],
       [ 1.06817596, -0.31351748,  0.19738551,  1.42738255, -1.1055416 ],
       [ 0.34931937, -0.37544419,  0.19738551,  1.15155984, -1.1055416 ],
       [-1.0883938 , -0.23274798,  0.19738551, -0.50337646,  0.90453403],
       [-0.90867965, -0.40869156,  0.19738551, -0.50337646, -1.1055416 ],
       [ 0.52903352, -0.44206738,  0.19738551,  1.15155984,  0.90453403],
       [-0.81882258, -0.45289808,  0.19738551, -0.77919917,  0.90453403],
       [ 1.33774718,  5.5623082 ,  0.19738551, -0.77919917,  0.90453403],
       [ 0.88846181, -0.46961724,  0.19738551, -0.77919917, -1.1055416 ],
       [ 1.15803303, -0.3514756 ,  0.19738551,  0.87573712,  0.90453403],
       [ 0.61889059,  0.95559216,  0.19738551, -0.50337646,  0.90453403],
       [ 0.97831888,  0.47191977,  0.19738551, -0.50337646,  0.90453403],
       [-0.81882258, -0.40661756,  0.19738551, -0.77919917,  0.90453403],
       [-0.99853673, -0.43380592,  0.19738551, -0.77919917,  0.90453403],
       [-0.90867965, -0.47313535,  0.19738551, -0.50337646, -1.1055416 ],
       [-0.72896551, -0.16479491,  0.19738551, -0.50337646,  0.90453403],
       [-0.72896551, -0.0125962 ,  0.19738551,  1.42738255, -1.1055416 ],
       [-0.90867965, -0.3151516 ,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.01010892, -0.4001779 ,  0.19738551, -0.77919917,  0.90453403],
       [ 0.2594623 , -0.36459752,  0.19738551, -0.50337646, -1.1055416 ],
       [-0.90867965,  1.22050344,  0.19738551,  1.15155984,  0.90453403],
       [-0.99853673, -0.29898869,  0.19738551,  1.15155984, -1.1055416 ],
       [ 0.61889059, -0.0982395 ,  0.19738551, -0.77919917, -1.1055416 ],
       [-1.0883938 ,  0.76523009,  0.19738551, -0.77919917,  0.90453403],
       [-0.99853673, -0.36907744,  0.19738551,  1.15155984,  0.90453403],
       [-0.54925136, -0.38457283,  0.19738551, -0.77919917,  0.90453403],
       [-0.36953721, -0.40262455,  0.19738551, -0.50337646, -1.1055416 ],
       [ 0.34931937,  0.19837808,  0.19738551,  1.15155984, -1.1055416 ],
       [ 1.06817596, -0.49411072,  0.19738551, -0.77919917, -1.1055416 ],
       [ 0.97831888,  2.48415619,  0.19738551, -0.50337646,  0.90453403],
       [-1.0883938 , -0.3054889 ,  0.19738551, -0.77919917, -1.1055416 ],
       [ 0.2594623 ,  0.05170642,  0.19738551, -0.50337646, -1.1055416 ],
       [-0.09996599, -0.18646878,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.01010892, -0.39570439,  0.19738551,  1.15155984,  0.90453403],
       [-1.17825087, -0.49390679,  0.19738551, -0.77919917, -1.1055416 ],
       [-0.54925136, -0.34604253,  0.19738551, -0.50337646, -1.1055416 ],
       [ 2.95517449, -0.45521686,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.36953721, -0.22607726,  0.19738551,  1.15155984,  0.90453403],
       [ 0.61889059, -0.42620289,  0.19738551, -0.77919917,  0.90453403],
       [-0.63910843, -0.39639881,  0.19738551,  1.15155984, -1.1055416 ],
       [ 0.70874766, -0.25008518,  0.19738551, -0.50337646,  0.90453403],
       [ 0.34931937, -0.34324426,  0.19738551, -0.77919917,  0.90453403],
       [ 0.70874766, -0.29074954,  0.19738551, -1.05502189, -1.1055416 ],
       [-0.54925136, -0.33386081,  0.19738551,  1.15155984, -1.1055416 ],
       [-0.63910843, -0.44824462, -5.06622805, -1.88249003,  0.90453403],
       [ 0.79860474, -0.00980244,  0.19738551,  1.15155984, -1.1055416 ]])
```

In [36]:
```python
X_test_scaled
```

Out[36]:
```
array([[ 0.61889059, -0.23597265,  0.19738551, -0.77919917,  0.90453403],
       [ 0.34931937, -0.38443136,  0.19738551,  0.87573712, -1.1055416 ],
       [ 1.87688962, -0.25930656,  0.19738551, -0.50337646,  0.90453403],
       [ 3.40445986, -0.23060312,  0.19738551,  1.42738255,  0.90453403],
       [ 0.52903352, -0.3171104 ,  0.19738551,  0.87573712,  0.90453403],
       [-0.09996599, -0.3194005 ,  0.19738551,  1.42738255, -1.1055416 ],
       [-1.17825087, -0.52912023,  0.19738551, -0.77919917, -1.1055416 ],
       [-0.27968014,  0.35284733,  0.19738551, -0.77919917,  0.90453403],
       [-0.81882258, -0.36725832,  0.19738551, -0.50337646,  0.90453403],
       [-0.27968014, -0.25436564,  0.19738551, -0.50337646,  0.90453403],
       [-0.72896551, -0.33761192,  0.19738551,  1.15155984,  0.90453403],
       [-0.27968014, -0.46081768,  0.19738551, -0.77919917,  0.90453403],
       [ 1.42760425, -0.30197255,  0.19738551,  1.15155984, -1.1055416 ],
       [ 1.06817596,  0.11532034,  0.19738551,  1.15155984,  0.90453403],
       [ 0.16960523,  0.88558398,  0.19738551, -0.77919917,  0.90453403],
       [ 0.97831888, -0.18734198,  0.19738551, -0.77919917,  0.90453403],
       [-1.0883938 , -0.25155268,  0.19738551, -0.50337646, -1.1055416 ],
       [-0.99853673, -0.21638028,  0.19738551,  1.42738255, -1.1055416 ],
       [-0.09996599, -0.54212812, -5.06622805, -1.88249003, -1.1055416 ],
       [ 1.33774718, -0.11951343,  0.19738551, -0.77919917, -1.1055416 ]])
```

Model Training

Using PyCaret

In [37]:
```python
exp_reg = setup(data = df2, target = 'annual', session_id=0, normalize=True, train_size=0.8,
        numeric_features=['months','weekspayment'],categorical_features=['gender_M'])
```

```
Setup Succesfully Completed.
```

	Description	Value
0	session_id	0
1	Transform Target	False
2	Transform Target Method	None
3	Original Data	(100, 6)
4	Missing Values	False
5	Numeric Features	4
6	Categorical Features	1
7	Ordinal Features	False
8	High Cardinality Features	False
9	High Cardinality Method	None
10	Sampled Data	(100, 6)
11	Transformed Train Set	(80, 6)
12	Transformed Test Set	(20, 6)
13	Numeric Imputer	mean
14	Categorical Imputer	constant
15	Normalize	True
16	Normalize Method	zscore
17	Transformation	False
18	Transformation Method	None
19	PCA	False
20	PCA Method	None
21	PCA Components	None
22	Ignore Low Variance	False
23	Combine Rare Levels	False
24	Rare Level Threshold	None
25	Numeric Binning	False
26	Remove Outliers	False
27	Outliers Threshold	None
28	Remove Multicollinearity	False
29	Multicollinearity Threshold	None
30	Clustering	False
31	Clustering Iteration	None
32	Polynomial Features	False
33	Polynomial Degree	None
34	Trignometry Features	False
35	Polynomial Threshold	None
36	Group Features	False
37	Feature Selection	False
38	Features Selection Threshold	None
39	Feature Interaction	False
40	Feature Ratio	False
41	Interaction Threshold	None

In [38]:
```python
compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
0	Random Forest	29301.5291	2163230089.6387	42515.7030	0.3529	0.3022	0.2724	0.1758
1	Extra Trees Regressor	30909.3561	2334604919.2863	43150.5733	0.3501	0.3118	0.2863	0.1388
2	AdaBoost Regressor	32635.1157	2127994922.1505	42943.7651	0.3479	0.3456	0.3309	0.0767
3	Gradient Boosting Regressor	33365.8582	2111731316.6444	43646.7740	0.3123	0.3464	0.3276	0.0417
4	Extreme Gradient Boosting	32658.4512	2493513976.4447	46506.4956	0.3076	0.3346	0.2863	0.0910
5	CatBoost Regressor	32752.6319	2563181216.0432	44846.5924	0.2778	0.3548	0.3265	0.7928
6	Elastic Net	38823.9344	3099121552.3342	50633.3856	0.2085	0.4152	0.4046	0.0053
7	Random Sample Consensus	36678.6640	3169982088.6563	51390.4456	0.2040	0.3875	0.3116	0.0752
8	Huber Regressor	36626.1631	3066209514.8814	50086.0571	0.1856	0.3800	0.3248	0.0100
9	Orthogonal Matching Pursuit	39774.7737	3078734214.2651	51049.8354	0.1016	0.4021	0.3714	0.0016
10	Decision Tree	38305.2540	3219692271.8766	54047.7257	0.0111	0.4014	0.3432	0.0031
11	Ridge Regression	37439.5700	3098883144.8375	49984.0027	-0.0341	0.3849	0.3508	0.0016
12	Lasso Regression	37472.6130	3115766385.0226	50108.2217	-0.0517	0.3853	0.3498	0.0069
13	Linear Regression	37472.8836	3115861068.1703	50108.8670	-0.0518	0.3853	0.3498	0.0069
14	Lasso Least Angle Regression	37563.1804	3132128731.3781	50300.1457	-0.0563	0.3853	0.3494	0.0057
15	Least Angle Regression	37564.0646	3132626967.2336	50303.1164	-0.0567	0.3853	0.3494	0.0031
16	Passive Aggressive Regressor	46532.2455	5527293694.2722	67966.6885	-0.0808	0.4866	0.3131	0.0116
17	Support Vector Machine	53994.5347	5864698168.7939	70794.2223	-0.1538	0.5781	0.5451	0.0062
18	Bayesian Ridge	55199.3106	5613319379.3464	69890.1957	-0.1657	0.5996	0.6424	0.0069
19	K Neighbors Regressor	41543.4890	3839643993.4508	56916.9658	-0.1670	0.4273	0.3905	0.0047
20	Light Gradient Boosting Machine	42654.6880	4078318142.5164	57905.7787	-0.1886	0.4003	0.3800	0.0272
21	TheilSen Regressor	46391.7065	5370217310.1379	65878.8083	-1.4292	0.4317	0.4083	0.5651

Out[38]:
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=-1, oob_score=False,
                      random_state=0, verbose=0, warm_start=False)
```

Simple Regression Model

In [39]:
```python
lr = LinearRegression()
```

In [40]:
```python
lr.fit(X_train_scaled, y_train)
```

Out[40]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [41]:
```python
lr_pred = lr.predict(X_test_scaled)
```

In [42]:
```python
lr_pred
```

Out[42]:
```
array([145019.66871768,  70166.40490483, 127494.26832432,  38474.65966601,
        73716.71187725,  49173.18928733, 146180.56661006, 155001.58108455,
       138113.88238077, 136931.33224313,  67170.98898244, 146612.28747075,
        54513.95717847,  64019.18015509, 158538.92082806, 143956.72181563,
       136841.75957281,  54146.26294868, 289980.84667239, 139453.3436494 ])
```

In [44]:
```python
y_test
```

Out[44]:
```
array([189774.52,  37716.64, 196860.56,  54242.24,  47671.  ,  60223.8 ,
        79959.36, 210848.04, 173096.04, 132011.36,  51100.92,  99658.52,
        66168.44, 134576.52, 141362.52,  91406.12, 206827.92,  72565.48,
       255366.8 , 128463.4 ])
```

In [45]:
```python
mse = mean_squared_error(y_test,lr_pred)
mse
```

Out[45]:
```
1742935754.6537423
```

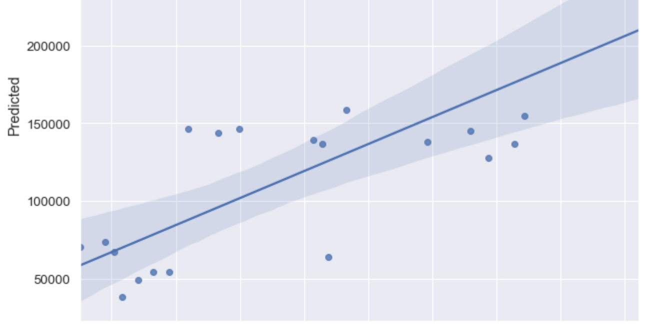
In [46]:
```python
rmse = np.sqrt(mse)
rmse
```

Out[46]:
```
41748.48206406722
```

In [47]:
```python
r2score = r2_score(y_test,lr_pred)
r2score
```

Out[47]:
```
0.5706038490094812
```

In [48]:
```python
fig, ax = plt.subplots(figsize=(10,8))
sns.regplot(x=y_test, y=lr_pred, ax=ax)
plt.title("Plot to compare actual vs predicted")
plt.ylabel("Predicted")
plt.xlabel("Actual")
plt.show()
```



Using XGBoost (Scikit-Learn)

Using RandomSearchCV

In [49]:
```python
model = XGBRegressor(random_state=0, n_estimators=100, objective='reg:squarederror')
```

In [50]:
```python
parameters = {'max_depth': np.arange(3,10,1),
              'learning rate': np.arange(0.05,0.3,0.05),
              'n_estimators':np.arange(100,1000,100),
              'min_child_weight': np.arange(1,4,1),
              'gamma':np.arange(0,10,2),
              'subsample':np.arange(0.5,0.9,0.1),
              'colsample_bytree':np.arange(0.5,0.9,0.1),
              'reg_alpha':np.arange(0,1,0.1),
              'reg_lambda':np.arange(0,1,0.1)
             }
```

In [51]:
```python
randm = RandomizedSearchCV(estimator=model, param_distributions = parameters, cv = 5, n_iter = 50,
                           n_jobs=-1, scoring='neg_mean_squared_error',random_state=0)
```

In [52]:
```python
randm.fit(X_train_scaled, y_train)
```

Out[52]:
```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=XGBRegressor(base_score=None, booster=None,
                                          colsample_bylevel=None,
                                          colsample_bynode=None,
                                          colsample_bytree=None, gamma=None,
                                          gpu_id=None, importance_type='gain',
                                          interaction_constraints=None,
                                          learning_rate=None,
                                          max_delta_step=None, max_depth=None,
                                          min_child_weight=None, missing=nan,
                                          monotone_constraints=None,
                                          n_...
                                        'min_child_weight': array([1, 2, 3]),
                                        'n_estimators': array([100, 200, 300, 400, 500, 600, 700, 80
0, 900]),
                                        'reg_alpha': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7,
0.8, 0.9]),
                                        'reg_lambda': array([0. , 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7,
0.8, 0.9]),
                                        'subsample': array([0.5, 0.6, 0.7, 0.8])},
                   pre_dispatch='2*n_jobs', random_state=None, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=0)
```

In [53]:
```python
randm.best_estimator_
```

Out[53]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.5, gamma=2, gpu_id=-1,
             importance_type='gain', interaction_constraints='',
             learning rate=0.15000000000000002, learning_rate=0.300000012,
             max_delta_step=0, max_depth=3, min_child_weight=1, missing=nan,
             monotone_constraints='()', n_estimators=800, n_jobs=0,
             num_parallel_tree=1, objective='reg:squarederror', random_state=0,
             reg_alpha=0.4, reg_lambda=0.1, scale_pos_weight=1, subsample=0.6,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

In [54]:
```python
randm.best_score_
```

Out[54]:
```
-2594910809.860133
```

In [55]:
```python
randm.best_params_
```

Out[55]:
```
{'subsample': 0.6,
 'reg_lambda': 0.1,
 'reg_alpha': 0.4,
 'n_estimators': 800,
 'min_child_weight': 1,
 'max_depth': 3,
 'learning rate': 0.15000000000000002,
 'gamma': 2,
 'colsample_bytree': 0.5}
```

Final Model

In [56]:
```python
xgbmodel = XGBRegressor(random_state=0, n_estimators=200, objective='reg:squarederror', subsample=0.7,
                        reg_lambda=0.4,reg_alpha=0.2,min_child_weight=1,max_depth=4,eta=0.25, gamma=4,
                        colsample_bytree=0.8)
```

In [57]:
```python
xgbmodel.fit(X_train_scaled,y_train,eval_set=[(X_test_scaled,y_test)],eval_metric='rmse',early_stopping
_rounds=10)
```

```
[0]	validation_0-rmse:108464.35156
Will train until validation_0-rmse hasn't improved in 10 rounds.
[1]	validation_0-rmse:87001.39062
[2]	validation_0-rmse:71516.21094
[3]	validation_0-rmse:62403.91797
[4]	validation_0-rmse:52160.75391
[5]	validation_0-rmse:45983.89453
[6]	validation_0-rmse:42712.09766
[7]	validation_0-rmse:40499.71484
[8]	validation_0-rmse:39906.14453
[9]	validation_0-rmse:39556.49609
[10]	validation_0-rmse:39097.35156
[11]	validation_0-rmse:40342.96875
[12]	validation_0-rmse:40113.82031
[13]	validation_0-rmse:39186.51562
[14]	validation_0-rmse:40743.67578
[15]	validation_0-rmse:41439.43359
[16]	validation_0-rmse:41451.07812
[17]	validation_0-rmse:41661.97656
[18]	validation_0-rmse:42399.53516
[19]	validation_0-rmse:42048.81250
[20]	validation_0-rmse:41679.58594
Stopping. Best iteration:
[10]	validation_0-rmse:39097.35156
```

Out[57]:
```
XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
             colsample_bynode=1, colsample_bytree=0.8, eta=0.25, gamma=4,
             gpu_id=-1, importance_type='gain', interaction_constraints='',
             learning_rate=0.25, max_delta_step=0, max_depth=4,
             min_child_weight=1, missing=nan, monotone_constraints='()',
             n_estimators=200, n_jobs=0, num_parallel_tree=1,
             objective='reg:squarederror', random_state=0, reg_alpha=0.2,
             reg_lambda=0.4, scale_pos_weight=1, subsample=0.7,
             tree_method='exact', validate_parameters=1, verbosity=None)
```

In [58]:
```python
y_pred = xgbmodel.predict(X_test_scaled)
```

In [59]:
```python
y_pred
```

Out[59]:
```
array([123855.125,  53267.34 ,  98036.33 ,  44264.766,  53234.176,
        60758.426, 116717.98 , 142476.62 , 127761.93 , 158373.56 ,
        55660.65 , 100023.43 ,  50072.734,  78182.79 , 143733.55 ,
       138132.31 , 169413.12 ,  67922.88 , 247472.95 , 115087.48 ],
      dtype=float32)
```

Model Evaluation

In [60]:
```python
mse = mean_squared_error(y_test,y_pred)
mse
```

Out[60]:
```
1528602724.8495464
```

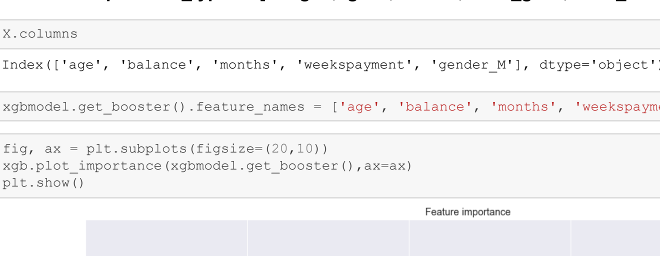
In [61]:
```python
rmse = np.sqrt(mse)
rmse
```

Out[61]:
```
39097.34933278146
```

In [62]:
```python
r2score = r2_score(y_test,y_pred)
r2score
```

Out[62]:
```
0.6234077333651278
```

In [63]:
```python
fig, ax = plt.subplots(figsize=(10,8))
sns.regplot(x=y_test, y=y_pred, ax=ax)
plt.title("Plot to compare actual vs predicted")
plt.ylabel("Predicted")
plt.xlabel("Actual")
plt.show()
```



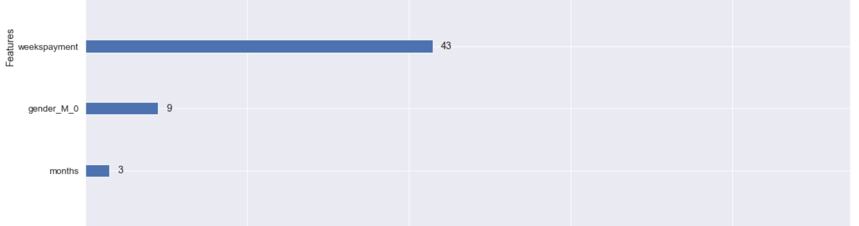
Available importance_types = [‘weight’, ‘gain’, ‘cover’, ‘total_gain’, ‘total_cover’]

In [64]:
```python
X.columns
```

Out[64]:
```
Index(['age', 'balance', 'months', 'weekspayment', 'gender_M'], dtype='object')
```

In [65]:
```python
xgbmodel.get_booster().feature_names = ['age', 'balance', 'months', 'weekspayment', 'gender_M_0']
```

In [66]:
```python
fig, ax = plt.subplots(figsize=(20,10))
xgb.plot_importance(xgbmodel.get_booster(),ax=ax)
plt.show()
```



Cross-Validation

In [67]:
```python
cv = cross_val_score(xgbmodel,X,y,cv=5,verbose=1,scoring='r2')
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   5 out of   5 | elapsed:    0.3s finished
```

In [68]:
```python
cv.mean()
```

Out[68]:
```
0.4714286293167129
```

Save the Model

In [69]:
```python
filename = 'anzmodel.sav'
dump(xgbmodel,open(filename,'wb'))
```

Conclusion: Tree based model is much better in accuracy than linear regression.

In [ ]: