In [30]:
```
import pandas as pd
import numpy as np
```

In [31]:
```
baseball = pd.read_table("http://jse.amstat.org/datasets/baseball.dat.txt", heade
                         names =["salary", "batting.avg", "OBP", "runs","hits", "
baseball.head()
```

Out[31]:

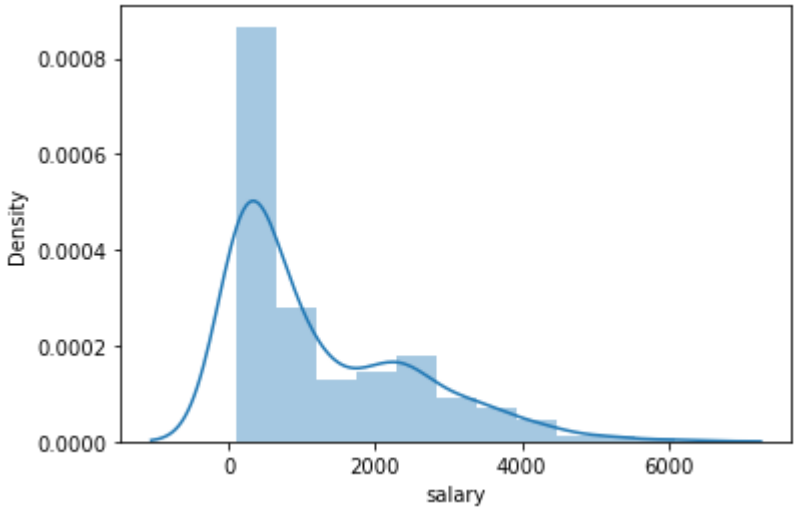
	salary	batting.avg	OBP	runs	hits	doubles	triples	homeruns	RBI	walks	strike.outs	stole
0	3300	0.272	0.302	69	153	21	4	31	104	22	80	
1	2600	0.269	0.335	58	111	17	2	18	66	39	69	
2	2500	0.249	0.337	54	115	15	1	17	73	63	116	
3	2475	0.260	0.292	59	128	22	7	12	50	23	64	
4	2313	0.273	0.346	87	169	28	5	8	58	70	53	

Q.no. 1. a.

In [32]:
```
import seaborn as sns
sns.distplot(baseball['salary'])
```

```
C:\Users\joshi\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

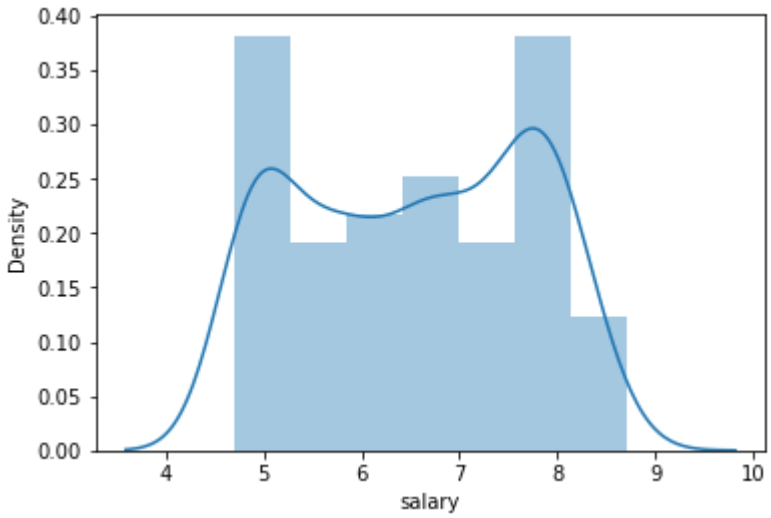
Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x172920f2e20>



In [33]:
```
baseball['salary'] = np.log(baseball['salary'])
sns.distplot(baseball['salary'])
```

```
C:\Users\joshi\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Futur
eWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function
with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

Out[33]: <matplotlib.axes._subplots.AxesSubplot at 0x17292157040>



The histogram for the salary of the players is skewed right with majority of players having salary less than 1000 and the histogram for the logarithmic salary, which is bimodal in nature, is more uniformly distributed than the variable salary alone.

Q.no.1. b.

In [34]:
```
baseball.isnull().sum()
```

Out[34]:
```
salary              0
batting.avg         0
OBP                 0
runs                0
hits                0
doubles             0
triples             0
homeruns            0
RBI                 0
walks               0
strike.outs         0
stolen.bases        0
errors              0
free.agency.elig    0
free.agent.91       0
arb.elig            0
arb.91              0
name                0
dtype: int64
```

In [35]:
```
baseball.dtypes
```

Out[35]:
```
salary              float64
batting.avg         float64
OBP                 float64
runs                  int64
hits                  int64
doubles               int64
triples               int64
homeruns              int64
RBI                   int64
walks                 int64
strike.outs           int64
stolen.bases          int64
errors                int64
free.agency.elig      int64
free.agent.91         int64
arb.elig              int64
arb.91                int64
name                 object
dtype: object
```

There are no missing values. There are five categorical variables (name, free.agency.elig, free.agent.91, arb.elig, and arb.91), three continous variables (salary, battingavg, and OBP), and 10 of the remaining variables (runs, hits, doubles, triples, homeruns, RBI, walks, strike.outs,

stolen.bases, and errors) are integer counts.

Q.no.2. a.

In [36]:
```
baseball = baseball.rename(columns = {"batting.avg":"battingavg", "strike.outs":"
                                      "stolen.bases":"stolenbases", "free.agency.
                                      "free.agent.91":"freeagent91", "arb.elig":"
import statsmodels.formula.api as smf
fit_full = smf.ols(formula='salary ~ battingavg + OBP + runs + hits + doubles + t
fit_full.summary()
```

Out[36]:

OLS Regression Results

Dep. Variable:	salary	**R-squared:**	0.802
Model:	OLS	**Adj. R-squared:**	0.792
Method:	Least Squares	**F-statistic:**	80.92
Date:	Tue, 26 Oct 2021	**Prob (F-statistic):**	4.63e-102
Time:	03:03:06	**Log-Likelihood:**	-259.75
No. Observations:	337	**AIC:**	553.5
Df Residuals:	320	**BIC:**	618.4
Df Model:	16		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	5.3951	0.257	20.976	0.000	4.889	5.901
battingavg	0.9018	2.097	0.430	0.667	-3.224	5.027
OBP	-2.1970	1.837	-1.196	0.233	-5.811	1.417
runs	0.0011	0.004	0.249	0.804	-0.007	0.010
hits	0.0049	0.003	1.931	0.054	-9.37e-05	0.010
doubles	-0.0024	0.007	-0.359	0.720	-0.015	0.011
triples	-0.0168	0.017	-1.007	0.315	-0.050	0.016
homeruns	0.0043	0.010	0.440	0.660	-0.015	0.023
RBI	0.0099	0.004	2.535	0.012	0.002	0.018
walks	0.0052	0.003	1.501	0.134	-0.002	0.012
strikeouts	-0.0055	0.002	-3.289	0.001	-0.009	-0.002
stolenbases	0.0047	0.004	1.289	0.198	-0.002	0.012
errors	-0.0079	0.006	-1.368	0.172	-0.019	0.003
freeagencyelig	1.5903	0.084	18.944	0.000	1.425	1.755
freeagent91	-0.2596	0.106	-2.440	0.015	-0.469	-0.050
arbelig	1.3183	0.091	14.417	0.000	1.138	1.498
arb91	-0.0753	0.187	-0.403	0.687	-0.443	0.293

Omnibus:	44.640	**Durbin-Watson:**	1.831
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	136.062

Skew:	-0.570	**Prob(JB):**	2.85e-30
Kurtosis:	5.896	**Cond. No.**	1.40e+04

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 1.4e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

==========

The above R-squared value of 0.802 or Residual is the training performance obtained using the data used to create the model.

Since the p-value of the F statistic is very small (4.63e-102), the model is valid (at least one variable is related to the response)

The p-values of RBI, strikeouts, freeagencyelig, freeagent91, and arbelig are meaningful as they are less than 0.05. However, since the p-value of the battingavg, OBP, runs, hits, doubles, triples, homeruns, walks, stolenbases, errors, and arb91 exceed 0.05, the null-hypothesis that "The respective variable is not related to the response" cannot be rejected. Therefore, it is not suitable to include the latter mentioned variables in the model.

Q.no.3. a.

In [37]:
```
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge, RidgeCV, Lasso, LassoCV
from sklearn.metrics import mean_squared_error
```

In [38]:
```
X = baseball.drop(['salary', 'name'], axis = 1)
y = baseball.salary
X_D0, X_D1 , y_D0, y_D1 = train_test_split(X, y, test_size=0.33, random_state=42)
```

Q.no.3. b.i.

In [39]:
```
alphas = 10**np.linspace(10,-2,100)*0.5
ridgecv = RidgeCV(alphas = alphas, cv= 10, scoring = 'neg_mean_squared_error', nc
ridgecv.fit(X_D0, y_D0)
ridgecv.alpha_

ridge = Ridge(alpha = ridgecv.alpha_, normalize = True)
ridge.fit(X_D0, y_D0)
```

Out[39]:
```
Ridge(alpha=0.026683496156031508, normalize=True)
```

Q.no.3. b.ii.

In [40]:
```
lassocv = LassoCV(alphas = alphas, cv = 10, max_iter = 100000, normalize = True)
lassocv.fit(X_D0, y_D0)
lassocv.alpha_

lasso = Lasso(alpha = lassocv.alpha_)
lasso.fit(X_D0, y_D0)
```

Out[40]:
```
Lasso(alpha=0.005)
```

Q.no.3. c.

In [41]:
```
alphas
```

Out[41]:
```
array([5.00000000e+09, 3.78231664e+09, 2.86118383e+09, 2.16438064e+09,
       1.63727458e+09, 1.23853818e+09, 9.36908711e+08, 7.08737081e+08,
       5.36133611e+08, 4.05565415e+08, 3.06795364e+08, 2.32079442e+08,
       1.75559587e+08, 1.32804389e+08, 1.00461650e+08, 7.59955541e+07,
       5.74878498e+07, 4.34874501e+07, 3.28966612e+07, 2.48851178e+07,
       1.88246790e+07, 1.42401793e+07, 1.07721735e+07, 8.14875417e+06,
       6.16423370e+06, 4.66301673e+06, 3.52740116e+06, 2.66834962e+06,
       2.01850863e+06, 1.52692775e+06, 1.15506485e+06, 8.73764200e+05,
       6.60970574e+05, 5.00000000e+05, 3.78231664e+05, 2.86118383e+05,
       2.16438064e+05, 1.63727458e+05, 1.23853818e+05, 9.36908711e+04,
       7.08737081e+04, 5.36133611e+04, 4.05565415e+04, 3.06795364e+04,
       2.32079442e+04, 1.75559587e+04, 1.32804389e+04, 1.00461650e+04,
       7.59955541e+03, 5.74878498e+03, 4.34874501e+03, 3.28966612e+03,
       2.48851178e+03, 1.88246790e+03, 1.42401793e+03, 1.07721735e+03,
       8.14875417e+02, 6.16423370e+02, 4.66301673e+02, 3.52740116e+02,
       2.66834962e+02, 2.01850863e+02, 1.52692775e+02, 1.15506485e+02,
       8.73764200e+01, 6.60970574e+01, 5.00000000e+01, 3.78231664e+01,
       2.86118383e+01, 2.16438064e+01, 1.63727458e+01, 1.23853818e+01,
       9.36908711e+00, 7.08737081e+00, 5.36133611e+00, 4.05565415e+00,
       3.06795364e+00, 2.32079442e+00, 1.75559587e+00, 1.32804389e+00,
       1.00461650e+00, 7.59955541e-01, 5.74878498e-01, 4.34874501e-01,
       3.28966612e-01, 2.48851178e-01, 1.88246790e-01, 1.42401793e-01,
       1.07721735e-01, 8.14875417e-02, 6.16423370e-02, 4.66301673e-02,
       3.52740116e-02, 2.66834962e-02, 2.01850863e-02, 1.52692775e-02,
       1.15506485e-02, 8.73764200e-03, 6.60970574e-03, 5.00000000e-03])
```

The tuning parameter, lambda, for both of the variable selection methods have been chosen by performing a 10-fold cross validation with different set of lambdas shown above. The one that resulted in most efficient prediction in the cross-validation was then selected to fit into the Ridge() and Lasso() models.

Q.no.3. d.

In [42]:
```
pd.Series(ridge.coef_, index = X.columns)
```

Out[42]:
```
battingavg       -0.026696
OBP              -1.193753
runs             -0.003033
hits              0.006111
doubles          -0.001098
triples          -0.010288
homeruns          0.003594
RBI               0.009204
walks             0.005465
strikeouts       -0.003281
stolenbases       0.001835
errors           -0.010493
freeagencyelig    1.612707
freeagent91      -0.405020
arbelig           1.283800
arb91            -0.083832
dtype: float64
```

In [43]:
```
pd.Series(lasso.coef_, index = X.columns)
```

Out[43]:
```
battingavg       -0.000000
OBP              -0.000000
runs             -0.007047
hits              0.007559
doubles          -0.003270
triples          -0.009876
homeruns          0.003859
RBI               0.010088
walks             0.005054
strikeouts       -0.002976
stolenbases       0.003518
errors           -0.011724
freeagencyelig    1.636456
freeagent91      -0.375027
arbelig           1.295248
arb91            -0.000000
dtype: float64
```

Q.no.3. e.

In [44]:
```
mean_squared_error(y_D1, ridge.predict(X_D1))
```

Out[44]:
```
0.32470580502716356
```

In [45]:
```
mean_squared_error(y_D1, lasso.predict(X_D1))
```

Out[45]:
```
0.3340127864886142
```

Q.no.4

In [46]:
```
alphas = 10**np.linspace(10,-2,100)*0.5
ridgecv = RidgeCV(alphas = alphas, cv= 10, scoring = 'neg_mean_squared_error', nc
ridgecv.fit(X, y)
ridgecv.alpha_

fit_final = Ridge(alpha = ridgecv.alpha_, normalize = True)
fit_final.fit(X, y)

pd.Series(fit_final.coef_, index = X.columns)
```

Out[46]:
```
battingavg        0.596579
OBP              -1.779626
runs              0.002238
hits              0.004542
doubles           0.000343
triples          -0.015405
homeruns          0.005968
RBI               0.008480
walks             0.004629
strikeouts       -0.004994
stolenbases       0.004018
errors           -0.007587
freeagencyelig    1.506393
freeagent91      -0.203964
arbelig           1.245256
arb91            -0.055978
dtype: float64
```

Both the models have larger weights for the same variables (battingavg, OBP, freeagencyelig, freeagent91, and arbelig) and none of the models eliminate any of the variables.