FIT1043 Introduction to Data Science Assignment 2

Ian WONG

Student ID: 30612616

Question 1

1. Introduction

a. Introduction

In Question 1, we will be looking at a dataset containing information about borrowers that have taken a loan from an investor. Using this dataset, we are trying to create a model to predict whether or not a borrower will pay back their loan in full, based on numerous characteristics of the borrower and the loan.

b. Importing Libraries + Reading Data

The first library we will be importing is **pandas**, which is used to create *DataFrames*. These dataframes allow us to manage and manipulate data more easily.

The second library we will be importing is **matplotlib.pyplot**, which is used to plot and visualise data in Python.

We will also be importing **numpy**, which is a package that is used for working with arrays.

In [3]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Using the `read_csv` command from **pandas**, we are reading the dataset from `load_data.csv` :

In [4]:
```
loan_data = pd.read_csv('Assignment 2Data/loan_data.csv')
```

To make sure we are reading the data in correctly, we are going to use `head()` , `tail()` , and `sample()` to check the data:

In [6]:
```
loan_data.head()
```

Out[6]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.ut
0	1	0.1189	829.10	11.350407	19.48	737	5639.958333	28854	52.
1	1	0.1071	228.22	11.082143	14.29	707	2760.000000	33623	76.

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revol.ut
2	1	0.1357	366.86	10.373491	11.63	682	4710.000000	3511	25.
3	1	0.1008	162.34	11.350407	8.10	712	2699.958333	33667	73.
4	1	0.1426	102.92	11.299732	14.97	667	4066.000000	4740	39.

In [7]:
```
loan_data.tail()
```

Out[7]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revo
9573	0	0.1461	344.76	12.180755	10.39	672	10474.000000	215372	
9574	0	0.1253	257.70	11.141862	0.21	722	4380.000000	184	
9575	0	0.1071	97.81	10.596635	13.09	687	3450.041667	10036	
9576	0	0.1600	351.58	10.819778	19.18	692	1800.000000	0	
9577	0	0.1392	853.43	11.264464	16.28	732	4740.000000	37879	

In [8]:
```
loan_data.sample(5)
```

Out[8]:

	credit.policy	int.rate	installment	log.annual.inc	dti	fico	days.with.cr.line	revol.bal	revo
6557	1	0.1183	728.95	12.254863	13.51	757	3814.041667	298	
49	1	0.0743	178.69	11.736069	0.27	792	7488.958333	2817	
4398	1	0.0740	186.36	11.608236	12.86	787	9484.000000	63621	
4082	1	0.1600	527.36	11.156251	7.83	662	6329.958333	18503	
332	1	0.1109	327.82	10.633449	19.32	702	2849.000000	15368	

In [6]:
```
loan_data.mean()
```

Out[6]:
```
credit.policy            0.804970
int.rate                 0.122640
installment            319.089413
log.annual.inc          10.932117
dti                     12.606679
fico                   710.846314
days.with.cr.line     4560.767197
revol.bal            16913.963876
revol.util              46.799236
inq.last.6mths           1.577469
delinq.2yrs              0.163708
pub.rec                  0.062122
not.fully.paid           0.160054
dtype: float64
```

c. Histogram

In this task, we are asked to create a stacked histogram for `not.fully.paid` against borrowers' credit scores `fico`.

Using `.groupby()` and `.agg()`, we are grouping the dataframe by `fico` and finding:

- the total number of loans; `count`
- the total number of fully paid loans; `sum`

In [9]:
```
groupbyFico = loan_data.groupby('fico')['not.fully.paid'].agg(['count','sum'])
```

Using the resulting columns, we can calculate the number of loans not fully paid and fully paid:

- `Not Fully Paid = sum`
- `Fully Paid = count - sum`

In [10]:
```
groupbyFico['Not Fully Paid'] = groupbyFico['sum']
groupbyFico['Fully Paid'] = groupbyFico['count'] - groupbyFico['sum']
```

Using `.loc`, we select the newly created `Not Fully Paid` and `Fully Paid` columns and use `.reset_index()` and `rename()` to clean up the data.

In [11]:
```
groupbyFico = groupbyFico.loc[:, ('Not Fully Paid', 'Fully Paid')]
groupbyFico = groupbyFico.reset_index()
groupbyFico.rename(
    columns = {'fico' : 'Fico'},
    inplace = True
)
groupbyFico.head()
```

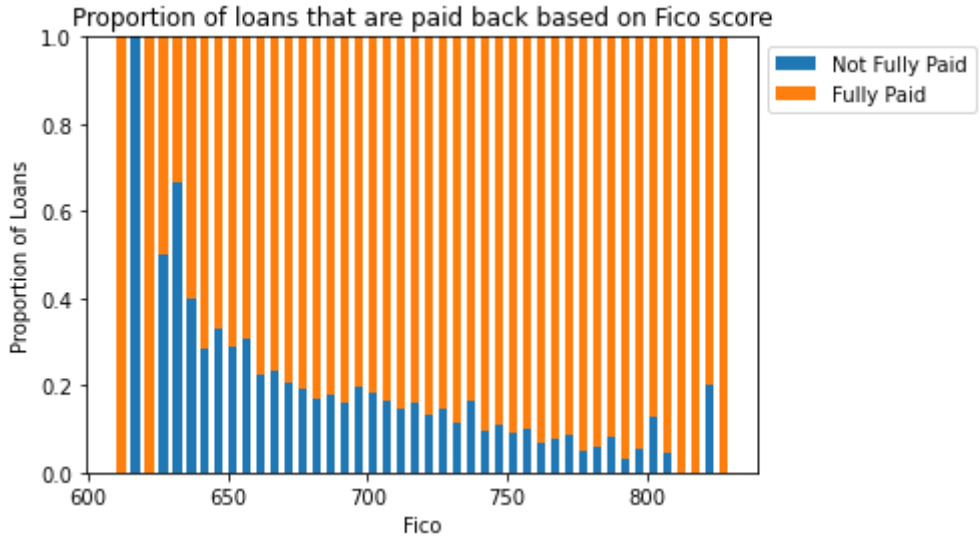
Out[11]:

	Fico	Not Fully Paid	Fully Paid
0	612	0	2
1	617	1	0
2	622	0	1
3	627	1	1
4	632	4	2

Using `plt.bar`, we are creating a stacked histogram with `fico` as the x axis and number of loans as the y axis.

We are doing this as a proportion of loans, as we wish to see whether `fico` score affects how likely a loan is to be paid off.

In [12]:
```
percent = groupbyFico['Not Fully Paid']/(groupbyFico['Not Fully Paid'] + groupbyFico

barWidth = 3

plt.bar(groupbyFico['Fico'], percent, width = barWidth, label = 'Not Fully Paid')
plt.bar(groupbyFico['Fico'], 1 - percent, bottom = percent, width = barWidth, label

plt.xlabel('Fico')
plt.ylabel('Proportion of Loans')

plt.legend(loc='upper left', bbox_to_anchor=(1,1), ncol=1)
plt.title('Proportion of loans that are paid back based on Fico score')

plt.show()
```



By observing the graph above, we can see that the proportion of fully paid loans increases as credit score, `fico`, increases.

This indicates that a borrower with a *higher* credit score is **more likely** to fully pay back a loan compared to a borrower with a *lower* credit score.

d. Fico and Interest Rate Plot

In this task, we are asked to create a plot to show the relationship between `fico` and `int.rate`.

We will be using a scatter plot to represent the data points and be using **LinearRegression** from the package `sklearn.linear_model` to obtain a line of best fit to better represent the trend of the data.

First, we will be importing **LinearRegression** from `sklearn.linear_model`:

In [13]:
```
from sklearn.linear_model import LinearRegression
```

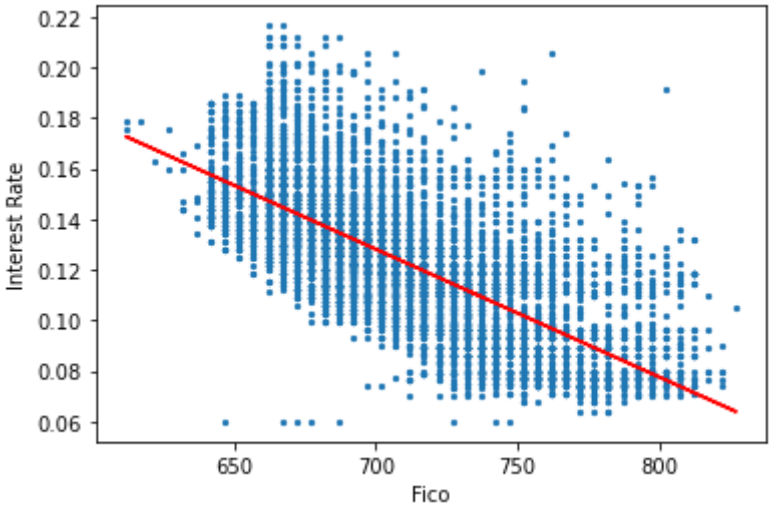
Using `np.array()`, we fit `fico` and `int.rate` into a numpy array as x and y variables respectively.

In [14]:
```
x = np.array(loan_data['fico'])
x = x[:, np.newaxis]
y = np.array(loan_data['int.rate'])
y = y[:, np.newaxis]
```

Usinng `plt.scatter()`, we create a scatter plot for `fico` and `int.rate` and fit a line of best fit using `model.fit()`.

In [15]:
```
model = LinearRegression()
model.fit(x,y)
y_pred = model.predict(x)

plt.scatter(x, y, s=5)
plt.plot(x, y_pred, color='r')
plt.xlabel('Fico')
```

```
plt.ylabel('Interest Rate')
plt.show()
```



By observing the scatter plot above and the predicted line of best fit, we can see there is a downward trend in the data, with interest rate decreasing as credit score increases.

This indicates that borrowers with *higher* credit scores are considered to be **less risky** and are therefore assigned *lower* interest rates.

2. Supervised Learning

a. Supervised ML, Labelled Data, and Training/Testing Datasets

Supervised Machine Learning: Supervised ML is when we are trying to teach a machine to predict/infer an output by feeding it labelled data and having it attempt to "*approximate the mapping function so well that when you have new input data, you can predict the output variable for that data*". (Taken from Week 5 Lecture)

Labelled Data: Labelled data is data that has an associated label, such as an ID, cateogorical type, or a number attached to it. It is through these labels in Supervised ML that a machine can draw patterns from the data and try to make inferences.

Training and Testing Datasets: Data is split into training and testing datasets. The Training Dataset is used to help the machine develop a predictive model by feeding it inputs and their corresponding outputs. The Testing Dataset is used to test how well the algorithm the machine created works, and helps to adjust it based on the results.

b. Labels and Features

The **label** in this dataset is `not.fully.paid` , as we are trying to identify how the features affect how likely a loan will be fully paid off.

Some **features** in this dataset include:

- `fico` ; the credit score of the borrower
- `dti` ; debt to income ratio
- `revol.bal` ; the borrower's revolving balance
- `revol.util` ; the borrower's utilisation rate

- `delinq.2yrs` ; number of times the borrower had been past due a payment
- `pub.rec` ; number of derogatory public records

These features are likely to tell the lender if the borrower is trustworthy or not.

c. Splitting the data

In this task, we are going to split our data into a training dataset containing 80% of `loan_data` and a testing dataset containing the remaining 20%.

We start off by using `.columns` to take a look at our dataset's columns:

In [16]:
```
loan_data.columns
```

Out[16]:
```
Index(['credit.policy', 'int.rate', 'installment', 'log.annual.inc', 'dti',
       'fico', 'days.with.cr.line', 'revol.bal', 'revol.util',
       'inq.last.6mths', 'delinq.2yrs', 'pub.rec', 'not.fully.paid'],
      dtype='object')
```

Then, using `.iloc` we are going to select the appropriate columns for the input data **x** and output data **y**:

In [17]:
```
X = loan_data.iloc[:, 0:11].values # input data
X
```

Out[17]:
```
array([[1.0000e+00, 1.1890e-01, 8.2910e+02, ..., 5.2100e+01, 0.0000e+00,
        0.0000e+00],
       [1.0000e+00, 1.0710e-01, 2.2822e+02, ..., 7.6700e+01, 0.0000e+00,
        0.0000e+00],
       [1.0000e+00, 1.3570e-01, 3.6686e+02, ..., 2.5600e+01, 1.0000e+00,
        0.0000e+00],
       ...,
       [0.0000e+00, 1.0710e-01, 9.7810e+01, ..., 8.2900e+01, 8.0000e+00,
        0.0000e+00],
       [0.0000e+00, 1.6000e-01, 3.5158e+02, ..., 3.2000e+00, 5.0000e+00,
        0.0000e+00],
       [0.0000e+00, 1.3920e-01, 8.5343e+02, ..., 5.7000e+01, 6.0000e+00,
        0.0000e+00]])
```

In [18]:
```
y = loan_data.iloc[:, 12].values # output data
y
```

Out[18]:
```
array([0, 0, 0, ..., 1, 1, 1], dtype=int64)
```

We are using `sklearn.model_selection.train_test_split` to split the data, using `test_size = 0.2` :

In [19]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.2, random_state=0
)
```

3. Classification

a. Binary and Multi-class Classification

In machine learning, Classification is the method of classifying data into different groups.

Binary classification is the categorisation of data into two distinct separate classes.

Multi-class Classfication is the categorisation of data into multiple different categories.

In this question, we are separating the data into two distinct categories of `not.fully.paid` : **0** or **1**. Therefore, our problem fits under the description of **Binary Classification**.

b. Training

Using `DecisionTreeClassifier` and `RandomForestClassifier` from `sklearn.tree` , we are going to train a **decision tree** and **random forest model**.

First, we need to normalise the data, using `StandardScaler()` . As the range of data can vary greatly, we scale the features down so they are proportionally similar:

In [20]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Decision Tree

Using `DecisionTreeClassifier` , we are going to fit a Decision Tree Classification to the training dataset:

In [21]:
```
# Fitting Decision Tree Classification to the Training set
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(
    criterion = 'entropy', random_state = 0
)
classifier.fit(X_train, y_train)
```

Out[21]:
```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

Random Forest

Using `RandomForestClassifier` , we are going to fit a Random Forest Classification to the training dataset:

In [22]:
```
# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
rfclassifier = RandomForestClassifier(
    n_estimators = 2000,
    criterion = 'entropy',
    random_state = 0
)
rfclassifier.fit(X_train, y_train)
```

Out[22]:
```
RandomForestClassifier(criterion='entropy', n_estimators=2000, random_state=0)
```

c. Prediction

i. Predict results

Decision Tree

Using `.predict`, we are predicting the test dataset for a Decision Tree Classification:

In [23]:
```
# Predicting the Test set results for Decision Tree Classification:
y_pred_dt = classifier.predict(X_test)
y_pred_dt
```

Out[23]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

Random Forest

Using `.predict`, we are predicting the test dataset for a Random Forest Classification:

In [24]:
```
# Predicting the Test set results for Random Forest Classification:
y_pred_rf = rfclassifier.predict(X_test)
y_pred_rf
```

Out[24]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

ii. Display the confusion matrix

Using `.confusion_matrix` from the `sklearn.metrics` package, we are creating a confusion matrix for both classifications.

In [25]:
```
from sklearn.metrics import confusion_matrix
```

Decision Tree Confusion Matrix:

In [26]:
```
# Making the Confusion Matrix for Decision Tree Classification
from sklearn.metrics import confusion_matrix
cm_dt = confusion_matrix(y_test, y_pred_dt)
cm_dt
```

Out[26]:
```
array([[1379,  236],
       [ 226,   75]], dtype=int64)
```

Random Forest Confusion Matrix:

In [27]:
```
# Making the Confusion Matrix for Random Forest Classifcation
from sklearn.metrics import confusion_matrix
cm_rf = confusion_matrix(y_test, y_pred_rf)
cm_rf
```

Out[27]:
```
array([[1609,    6],
       [ 296,    5]], dtype=int64)
```

4. Conclusion

Based on the confusion matrices from 3.c.ii, we can see that the **accuracy** of Random Forest Classification is far greater than Decision Tree Classification in this problem, with RF Classification having an accuracy of **84%** `(1609 + 5 / 1609 + 6 + 296 + 5)` versus DT Classification's accuracy of **76%** `(1379 + 75 / 1379 + 236 + 226 + 75)`.

Another metric that is of interest is **sensitivity**. That is, when the actual value is positive (a loan will not be fully paid back), how often is the prediction correct? In this case, DT Classification has a lower sensitivity when compared to RF, with DT having a precision of **85.4%** versus RF's sensitivity of **99.6%**.

In this problem, false positives (a loan that is predicted to be not be fully paid is paid in full) are more acceptable that false negatives (a loan that is predicted to be paid in full is not fully paid). Therefore, we should be optimising for sensitivity.

RF Classification is performing better than DT Classification in my opinion because of it's higher accuracy and sensitivity in this problem.

Question 2.

1. Introduction

a. Introduction

In this question, we are acting as a data scientist in an E-Commerce company. The company has two ways that they sell clothing online:

- through their **website**
- through their **mobile app**

The company also offers style/clothing advice sessions indoors. Customers attend these sessions and then they order their clothes on either the website or mobile app.

The company is asking you to determine whether they should focus their efforts on the mobile app or the website.

b. Importing Libraries + Reading Data

The first library we will be importing is **pandas**, which is used to create *DataFrames*. These dataframes allow us to manage and manipulate data more easily.

The second library we will be importing is **matplotlib.pyplot**, which is used to plot and visualise data in Python.

We will also be importing **numpy**, which is a package that is used for working with arrays.

In [28]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

Using `.read_csv()`, we are going to read in the dataset `customers-shop.csv`:

In [29]:
```
shop_data = pd.read_csv('Assignment 2Data/customers-shop.csv')
```

To make sure we are reading the data in correctly, we are going to use `head()`, `tail()`, and `sample()` to check the data:

In [30]:
```
shop_data.head()
```

Out[30]:

	Customer info-color Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
0	Violet	34.497268	12.655651	39.577668	4.082621	587.951054
1	DarkGreen	31.926272	11.109461	37.268959	2.664034	392.204933
2	Bisque	33.000915	11.330278	37.110597	4.104543	487.547505
3	SaddleBrown	34.305557	13.717514	36.721283	3.120179	581.852344
4	MediumAquaMarine	33.330673	12.795189	37.536653	4.446308	599.406092

In [31]:
```
shop_data.tail()
```

Out[31]:

	Customer info-color Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
495	Tan	33.237660	13.566160	36.417985	3.746573	573.847438
496	PaleVioletRed	34.702529	11.695736	37.190268	3.576526	529.049004
497	Cornsilk	32.646777	11.499409	38.332576	4.958264	551.620146
498	Teal	33.322501	12.391423	36.840086	2.336485	456.469510
499	DarkMagenta	33.715981	12.418808	35.771016	2.735160	497.778642

In [32]:
```
shop_data.sample(5)
```

Out[32]:

	Customer info-color Avatar	Avg. Session Length	Time on App	Time on Website	Length of Membership	Yearly Amount Spent
389	LightCyan	34.195508	12.664193	37.027150	4.330407	581.308933
403	OliveDrab	33.085298	13.093537	38.315648	4.750360	632.123588
152	CornflowerBlue	32.510218	10.984836	37.396497	5.391275	555.892595
28	LemonChiffon	33.110205	11.982045	35.293088	3.923489	529.537665
259	CadetBlue	32.096109	10.804891	37.372762	2.699562	375.398455

c. Website Scatterplot

In this task, we are asked to create a scatter plot between **Time on Website** and **Yearly Amount Spent** and report on their correlation.

In order to do this, we are going to import `scipy.stats.linregress()` in order to get the correlation value and other values:

In [33]:
```
from scipy.stats import linregress
```

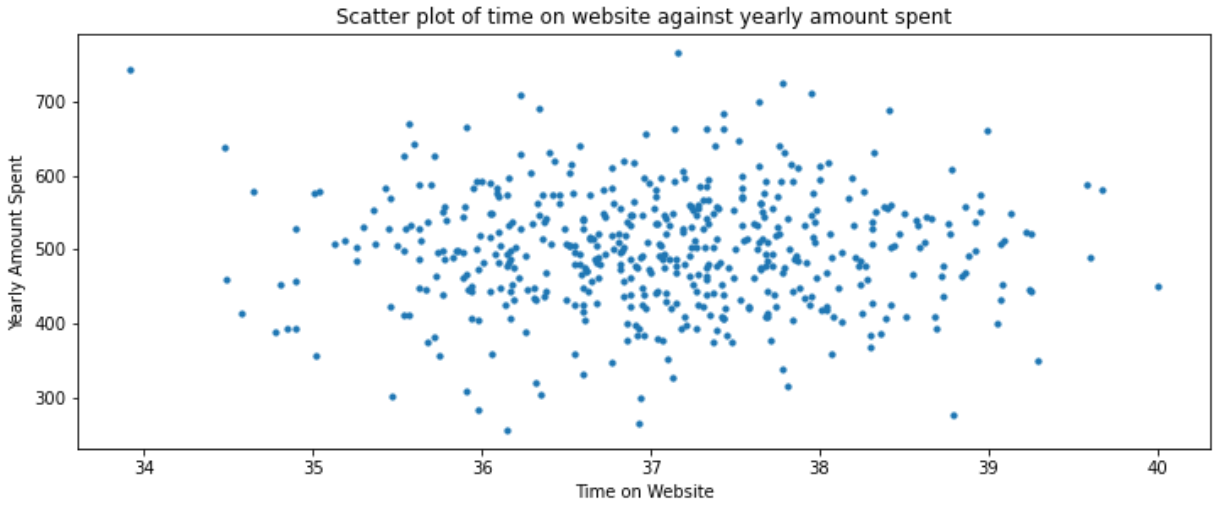
We are going to select the appropriate columns, with **Time on Website** as our x variable and **Yearly Amount Spent** as our y variable:

In [34]:

```
x = shop_data['Time on Website']
y = shop_data['Yearly Amount Spent']
```

Next, we are using `.scatter` for a scatter plot and `.figure` to increase the figure size so that we can see the data more clearly.

In [35]:
```
plt.figure(figsize=[12,4.5])
plt.scatter(x, y, s=10)
plt.xlabel('Time on Website')
plt.ylabel('Yearly Amount Spent')
plt.title('Scatter plot of time on website against yearly amount spent')
plt.show()
```



Taking a look at the scatter plot above, we can see that there is no evident correlation between time on website and yearly amount spent, with the data points being spread everywhere.

Using `linregress()`, we return 5 values:

- the **slope**,
- the **intercept**,
- the **r-value**,
- the **p-value**,
- and the **standard error**

We are only interested in the r-value for now, which is the correlation value:

In [36]:
```
slope, intercept, r_value, p_value, std_err = linregress(x,y)
r_value
```

Out[36]:
```
-0.00264084471796325
```

As we can see by the **r_value = -0.003**, there is a very slight negative correlation between time spent on the website and the yearly amount spent. However, it is very close to 0, that it is almost negligible.

d. App Scatter Plot

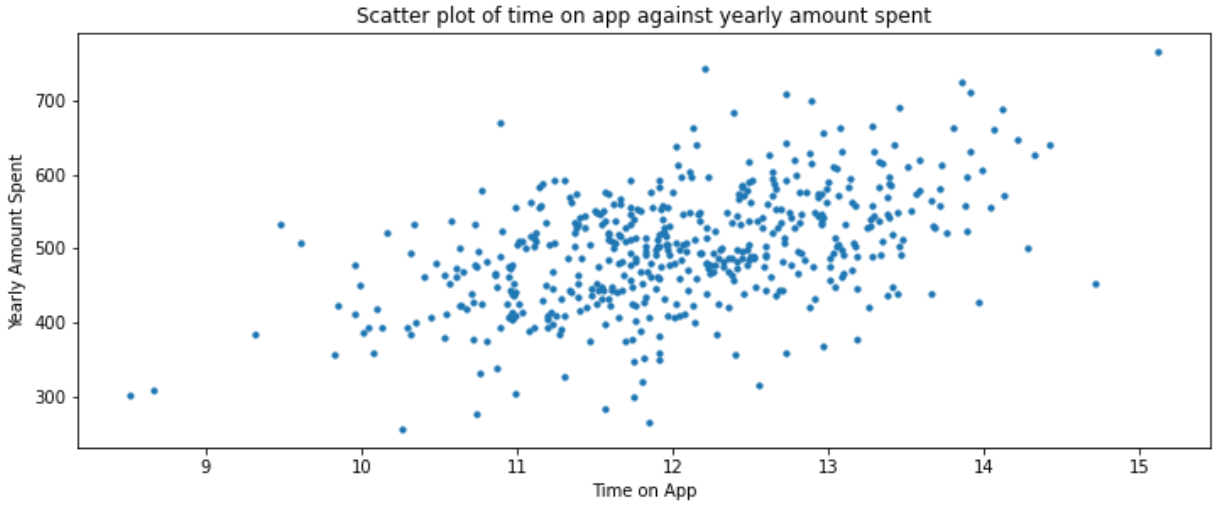
In this task, we are asked to create a scatter plot between **Time on App** and **Yearly Amount Spent** and report on their correlation.

First, we are going to select the appropriate columns, with **Time on App** as our x variable and **Yearly Amount Spent** as our y variable:

In [37]:
```
x = shop_data['Time on App']
y = shop_data['Yearly Amount Spent']
```

Next, we are using `.scatter` to create a scatter plot of the data:

In [38]:
```
plt.figure(figsize=[12,4.5])
plt.scatter(x, y, s=10)
plt.xlabel('Time on App')
plt.ylabel('Yearly Amount Spent')
plt.title('Scatter plot of time on app against yearly amount spent')
plt.show()
```



As we can see from the graph above, there appears to be a positive correlation between **Time on App** and **Yearly Amount Spent**. Therefore, as the customer spends more time on the app, they are likely to spend more on a yearly basis.

Using `linregress()`, we are obtaining the correlation value **r_value**:

In [39]:
```
slope, intercept, r_value, p_value, std_err = linregress(x,y)
r_value
```

Out[39]: `0.49932777005983486`

As we can see, the correlation value between **Time on App** and **Yearly Amount Spent** is about **0.5**. As the correlation value lies between -1 and 1, this is a relatively large positive correlation. Therefore, as the customer spends more time on the mobile, they are expected

Comparing this correlation with part c., it appears that the amount of time a customer spends on the mobile app affects the amount they spend yearly more compared to the time they spend on the website.

2. Supervised Learning

a. Separation of Features and Label

The **label** in this dataset is `Yearly Amount Spent`, as we are trying to identify how the features affect the amount a customer spends yearly.

The **features** in this dataset are:

- `Time on App`; the amount of time the customer has spent on the mobile app
- `Time on Website`; the amount of time the customer has spent on the website

b. Splitting the data

In this task, we are going to split our data into a training dataset containing 70% of `loan_data` and a testing dataset containing the remaining 30%.

We start off by using `.columns` to take a look at our dataset's columns:

In [40]:
```
shop_data.columns
```

Out[40]:
```
Index(['Customer info-color Avatar', 'Avg. Session Length', 'Time on App',
       'Time on Website', 'Length of Membership', 'Yearly Amount Spent'],
      dtype='object')
```

Using `.iloc`, we are selecting the appropriate columns for the x and y variables:

In [41]:
```
X = shop_data.iloc[:, [2, 3]].values
X
```

Out[41]:
```
array([[12.65565115, 39.57766802],
       [11.10946073, 37.26895887],
       [11.33027806, 37.11059744],
       [13.71751367, 36.72128268],
       [12.79518855, 37.5366533 ],
       [12.02692534, 34.47687763],
       [11.36634831, 36.68377615],
       [12.35195897, 37.37335886],
       [13.38623528, 37.53449734],
       [11.81412829, 37.14516822],
       [13.33897545, 37.22580613],
       [11.584783  , 37.08792607],
       [10.9612984 , 37.42021558],
       [12.95922609, 36.1446667 ],
       [13.14872569, 36.61995708],
       [12.63660605, 36.21376309],
       [11.73386169, 34.89409275],
       [12.01319469, 38.38513659],
       [14.71538754, 38.24411459],
       [13.98959256, 37.1905038 ],
       [11.36549203, 37.60779252],
       [12.8779837 , 37.44102134],
       [13.37856278, 38.73400629],
       [11.65757592, 36.77260376],
       [12.8936695 , 37.63575588],
       [11.76581265, 37.73852495],
       [12.78389178, 36.43064962],
       [13.00781942, 37.85177917],
       [11.98204499, 35.29308775],
       [11.96502   , 37.27781174],
       [12.30541781, 36.16364817],
       [10.86916381, 35.62244242],
       [13.38749211, 35.69417499],
       [13.10450724, 38.87804051],
       [11.63489325, 35.36862633],
       [11.9363865 , 38.768641  ],
```

```
[11.7550237 , 36.76572236],
[11.98441752, 37.0443614 ],
[ 9.95497597, 37.38831487],
[12.48901322, 36.37147981],
[11.73310622, 37.53429101],
[10.63456132, 37.49669015],
[12.50752537, 37.14286198],
[11.52987821, 36.88808605],
[13.41493474, 36.11243501],
[12.17052542, 39.13109673],
[13.14655143, 37.33544589],
[12.98851015, 36.46200326],
[11.86412636, 36.58272777],
[11.39806419, 36.5944567 ],
[13.85806246, 37.78026469],
[10.95679097, 37.26687826],
[10.32011626, 37.4534051 ],
[ 9.9845144 , 35.9334493 ],
[12.64519514, 38.46832111],
[11.58865542, 35.25224202],
[13.76153285, 39.25293095],
[10.56829469, 36.17312563],
[11.83228622, 36.81401056],
[12.06415663, 37.27122169],
[12.4955916 , 38.05260975],
[ 9.60731469, 36.49399255],
[13.72862718, 37.99702801],
[11.67006592, 37.40874848],
[13.27631301, 36.60077705],
[15.12699429, 37.15762409],
[12.69578975, 35.35844431],
[11.83547609, 36.37506611],
[11.30623234, 37.68040323],
[11.18753891, 40.00518164],
[11.88749413, 36.2650007 ],
[12.22893471, 36.15719115],
[10.67465347, 38.00658318],
[12.81711309, 37.03153922],
[13.58780608, 38.26035344],
[12.23805735, 38.73086174],
[11.55182117, 36.62883429],
[11.43337993, 35.89243163],
[10.88956686, 38.21257083],
[11.92884209, 36.91463331],
[12.59567131, 39.60037647],
[10.94725858, 35.88399439],
[13.89808199, 37.05891282],
[10.87555955, 37.78114256],
[10.3380727 , 36.15725594],
[11.63466822, 36.18253928],
[12.51766629, 37.15192066],
[10.60772387, 36.81909551],
[12.8288934 , 36.95161668],
[12.06881608, 36.1050005 ],
[12.53035737, 37.8752191 ],
[13.5162843 , 36.77312349],
[11.66226343, 36.05024078],
[13.66474788, 37.72438616],
[11.83023109, 36.6338567 ],
[13.29114305, 38.63362565],
[11.94234087, 38.06341359],
[13.34991294, 37.82739423],
[12.02011209, 39.07440021],
[12.08409173, 35.89035796],
[13.41075918, 35.99048895],
[11.28193107, 37.38531755],
[11.44890154, 37.58019043],
[12.59542035, 36.2620318 ],
[10.99422392, 38.07445242],
```

```
[13.08550576, 35.84582711],
[12.19047429, 36.15246209],
[10.71914974, 37.71250864],
[12.93155027, 38.16643556],
[13.45212896, 38.50300885],
[12.14937549, 37.32533422],
[12.17833133, 37.71598618],
[11.01048213, 38.41504158],
[11.60253219, 37.30968877],
[12.73221159, 35.60082055],
[11.22336889, 37.69230086],
[12.01102188, 36.70105232],
[11.7259101 , 35.9990993 ],
[12.08930957, 38.3099079 ],
[13.06863858, 37.54052036],
[12.85499037, 35.00748226],
[11.56402237, 37.67321037],
[11.1133299 , 37.38794554],
[12.80988347, 36.54966799],
[11.76117233, 37.57016384],
[12.57989417, 37.09326487],
[11.95792306, 36.63465233],
[10.65179378, 37.14600653],
[12.75207661, 36.71413784],
[11.54083244, 37.52642103],
[11.92439492, 37.24503247],
[12.42413041, 38.94882503],
[11.81058676, 37.41413357],
[12.75916898, 36.59911235],
[11.85189083, 37.42454792],
[12.70368793, 36.10091445],
[12.21525242, 36.59436168],
[11.281445  , 36.49440648],
[10.73536292, 37.45837473],
[10.96313178, 37.32728269],
[11.73509455, 36.59937399],
[12.41896198, 35.97765171],
[10.53730754, 35.7305524 ],
[11.9192424 , 39.29404346],
[11.91141556, 38.2747022 ],
[12.488067  , 36.51838359],
[12.38069498, 37.23200331],
[10.48050683, 37.33866962],
[12.29651768, 36.95155521],
[10.8616042 , 36.58443762],
[10.71635514, 38.30720401],
[12.38718417, 37.4311591 ],
[10.98483589, 37.39649748],
[12.96576148, 36.9663889 ],
[12.05026723, 36.95964319],
[13.45772494, 37.23880567],
[12.44304789, 37.32784804],
[12.20729849, 33.91384725],
[11.58631953, 39.09462703],
[11.37808709, 38.30447119],
[12.3643416 , 38.03910939],
[12.39943608, 35.01280603],
[12.54248105, 38.3113649 ],
[13.28043224, 36.93615938],
[11.79588668, 37.65861691],
[12.71803917, 37.66110668],
[12.03964784, 38.92408705],
[12.47445545, 35.03785616],
[13.16002004, 36.40774745],
[11.05232365, 37.63300942],
[10.63676108, 37.57883524],
[13.44340599, 36.87831537],
[13.28303287, 35.90729843],
[11.7477317 , 36.93988205],
```

```
[10.85960853, 38.83567008],
[11.69168613, 37.48091198],
[11.54876144, 38.57651555],
[11.72400223, 36.81385766],
[12.17857308, 35.67425603],
[12.31984506, 37.81915511],
[11.08436084, 37.95968432],
[13.17777453, 38.85604171],
[12.83280284, 37.67924462],
[11.509048  , 37.25305774],
[11.85766344, 36.08693433],
[12.29336584, 37.0646212 ],
[10.93325228, 36.54550623],
[13.33283928, 37.96439033],
[10.90255623, 36.09424195],
[12.27698171, 38.23260623],
[12.03880824, 37.63529941],
[11.72447386, 37.1531516 ],
[11.2026699 , 35.49396408],
[13.37806333, 36.33780003],
[11.38861262, 37.90913872],
[10.77107406, 37.27863979],
[11.81857176, 37.10203133],
[13.80879868, 37.42676892],
[11.65983284, 37.28139283],
[13.10010954, 35.90772143],
[13.06789568, 36.67822227],
[12.49432293, 36.04545942],
[11.34003593, 37.03951365],
[11.19966096, 38.6887091 ],
[11.33248778, 35.4598628 ],
[13.9194944 , 37.95201319],
[11.03135834, 38.25297827],
[11.08458409, 36.77601658],
[10.54264542, 35.5338635 ],
[11.79779551, 37.7773658 ],
[12.44261655, 38.13171203],
[12.10454243, 36.05964598],
[12.37849022, 38.76429717],
[13.72245368, 35.77311639],
[12.72590932, 36.5446641 ],
[13.6851189 , 34.89198327],
[10.01258337, 38.35495987],
[11.43553387, 36.22355746],
[12.35460708, 37.12234521],
[10.74853366, 35.73870747],
[11.59187167, 37.7436197 ],
[11.23650676, 37.67502073],
[11.10945633, 38.58585476],
[12.11494496, 36.28872395],
[11.86648121, 37.71777057],
[10.25654903, 36.14390846],
[12.16859647, 37.07361617],
[13.08535689, 37.6056528 ],
[13.01337563, 36.65127792],
[10.98397673, 37.95148946],
[13.10515852, 35.57484167],
[12.14474854, 37.25803146],
[13.90991551, 37.79223754],
[11.39520943, 37.33281446],
[13.03356618, 37.07679516],
[11.62277717, 35.96889569],
[12.6007504 , 37.37011822],
[13.18791099, 37.06708997],
[13.26676035, 36.9711951 ],
[11.75234317, 38.57360523],
[11.56811634, 36.90937821],
[12.63857212, 36.09722093],
[11.38864458, 39.08156475],
```

```
[11.8229833 , 36.94612578],
[12.72567737, 35.96566746],
[11.20115977, 37.68933698],
[11.34726361, 36.32365247],
[14.22097911, 37.52319665],
[11.54627576, 36.94795369],
[11.91763618, 36.84473382],
[12.27605698, 37.19279353],
[10.13171246, 34.84561239],
[12.02694222, 36.13389429],
[13.1722875 , 36.19975347],
[12.32629139, 36.67387836],
[13.89131342, 39.22071295],
[12.93092854, 36.3602472 ],
[11.37925718, 36.63610412],
[14.06938234, 38.99332245],
[10.80489056, 37.37276215],
[14.42649105, 37.3741835 ],
[13.04124459, 36.65520808],
[11.7397438 , 36.85481082],
[12.49105874, 38.23894085],
[12.89237451, 36.5273883 ],
[11.66886651, 37.34126616],
[11.77777204, 37.97982686],
[12.13879388, 36.85388246],
[12.22296745, 36.82275323],
[13.4023319 , 37.29204471],
[12.95627661, 38.65509454],
[12.88412462, 36.22604169],
[13.3254691 , 36.76860309],
[12.15858523, 36.57513378],
[11.22654566, 35.66993517],
[12.46113544, 37.74560774],
[ 9.84612491, 36.87631323],
[13.32541219, 36.89729461],
[13.67724584, 37.7446997 ],
[11.64496955, 37.02687697],
[13.48500899, 37.55088041],
[11.62099651, 38.41946853],
[13.01445915, 37.78903635],
[11.46698422, 35.67572763],
[11.68490419, 38.7170763 ],
[12.09396636, 36.62077386],
[14.32565494, 35.72182733],
[12.91484663, 39.06886445],
[ 9.82440177, 35.74277913],
[12.80775183, 38.55103029],
[13.05827793, 37.26387556],
[10.5345535 , 37.03479129],
[11.14343308, 35.94639915],
[10.98280553, 34.81063145],
[11.15396605, 37.24032955],
[11.94717532, 36.19083324],
[11.470565  , 37.06168877],
[11.03785041, 38.61733445],
[11.1673569 , 35.62658733],
[11.56293625, 35.97656497],
[12.22772828, 36.98591348],
[13.18681287, 38.0669296 ],
[13.39445179, 37.80697767],
[11.85139874, 36.92504304],
[12.26650384, 36.57503098],
[13.45922229, 36.33952101],
[11.79297182, 36.25781912],
[10.98574015, 37.36839124],
[12.63755716, 36.51708576],
[ 8.50815218, 35.46240008],
[11.65659203, 36.54860515],
[11.96689808, 36.54759628],
```

```
[11.80298578, 36.31576315],
[12.67740144, 35.62253059],
[13.03253498, 37.87095205],
[12.44761745, 37.53453024],
[12.23565925, 37.27757338],
[10.97316208, 36.60950715],
[13.66576979, 36.90022076],
[13.39112018, 37.19419105],
[12.96832561, 38.29611037],
[12.96889313, 37.33310742],
[10.73213134, 36.14579171],
[10.95235338, 37.64629179],
[12.60888879, 37.22939452],
[13.03951103, 36.31272657],
[10.99968366, 38.44276669],
[13.27895623, 37.38718053],
[10.62794923, 38.04031367],
[13.18518117, 35.92159519],
[11.35104901, 37.08884658],
[12.95481145, 37.10881638],
[11.73704064, 37.93518905],
[11.8873453 , 35.86244708],
[10.75713093, 36.59586795],
[11.61265077, 39.2488039 ],
[11.97906148, 38.26906069],
[12.46114744, 37.42899737],
[10.07946345, 38.07066426],
[12.58924056, 37.33224075],
[14.28801459, 36.77386137],
[11.9171157 , 37.76668677],
[11.40964462, 35.77778217],
[12.81539265, 37.95780983],
[11.67322925, 37.84065508],
[10.61053652, 37.97738874],
[12.50654819, 35.82346667],
[11.30446231, 37.83397173],
[10.88692118, 34.89782769],
[12.09588949, 36.37750903],
[12.69266143, 37.33359061],
[11.38677555, 38.19748325],
[10.72841854, 36.88119241],
[11.93689516, 35.90025278],
[12.7179951 , 35.12882235],
[10.96980287, 35.97457811],
[13.42054574, 37.7636904 ],
[11.85468192, 37.49189221],
[11.91885971, 35.71626916],
[13.14966956, 37.6504002 ],
[12.6853939 , 36.04898625],
[10.34787695, 39.0451557 ],
[ 9.3162892 , 36.91495155],
[12.58154773, 35.4442647 ],
[11.76444759, 37.92270381],
[12.64420212, 38.00182747],
[11.83211223, 36.84149164],
[10.44123506, 35.9389625 ],
[12.72971951, 36.2321098 ],
[ 9.47777761, 37.90601452],
[10.16317906, 37.76304108],
[11.60899794, 38.11045691],
[11.26825923, 36.95696543],
[11.16315954, 37.08831936],
[10.1016322 , 38.04345265],
[12.76153128, 36.90818962],
[12.05534016, 37.68546549],
[11.93593497, 35.78392374],
[12.45120001, 36.66579137],
[12.28446707, 38.29572526],
[12.78217179, 35.55077227],
```

```
[12.96030713, 37.95194632],
[ 9.95399501, 37.34573893],
[13.13002166, 35.42933439],
[11.52056679, 36.1891318 ],
[10.97255431, 34.57402759],
[13.53191346, 38.95246252],
[12.03950237, 34.48718475],
[10.94206962, 36.1704942 ],
[12.66419262, 37.02715036],
[11.73066139, 36.88214908],
[11.91867031, 37.31770495],
[11.51494907, 37.1280395 ],
[12.4181132 , 36.1553362 ],
[13.88727541, 38.38195597],
[12.84649907, 37.86921694],
[10.88982828, 35.56543624],
[12.91457009, 36.04620352],
[11.56852664, 38.91874854],
[11.97175079, 37.19936753],
[13.07869244, 37.32981863],
[11.95642648, 36.51734612],
[13.25273709, 37.30596142],
[13.09353728, 38.31564796],
[12.52747174, 36.68836654],
[11.23596895, 37.05261631],
[12.07483017, 35.56917032],
[12.04533198, 38.50588333],
[11.23074331, 36.99529017],
[11.9078441 , 35.18912244],
[13.05221039, 38.7756653 ],
[13.00436201, 36.98504142],
[11.52352268, 35.93804523],
[10.31471792, 36.7290294 ],
[12.43312853, 37.6269068 ],
[10.74518855, 38.79123469],
[11.01675564, 37.63731096],
[12.21685469, 36.9539597 ],
[12.32914702, 37.07437107],
[10.80696607, 36.01231723],
[12.1351014 , 37.14209369],
[12.94155573, 36.72527683],
[13.30029936, 36.39368386],
[10.39845771, 36.683393  ],
[11.88780042, 37.86144726],
[11.5914397 , 36.45689841],
[13.27147504, 37.23984661],
[12.9882206 , 39.67259096],
[14.03986726, 37.02226899],
[11.90650776, 38.42286529],
[11.1371403 , 38.40137369],
[12.90266489, 36.61119878],
[12.38651626, 35.63211222],
[11.82272175, 36.30854529],
[10.98576379, 36.35250277],
[14.13289346, 37.02347924],
[11.91221048, 36.08964366],
[11.48158715, 39.24096484],
[10.04731474, 37.18144731],
[12.42873693, 37.30536188],
[11.96598027, 36.83153585],
[11.12136606, 36.97937237],
[ 8.66834952, 35.90675637],
[12.50421981, 37.64583879],
[12.13250911, 35.45679815],
[11.73299146, 35.63395395],
[11.94659088, 36.48632507],
[12.55610762, 37.80550943],
[12.48070152, 37.68028761],
[12.59419407, 37.68387537],
```

```
       [10.94684194, 37.64780806],
       [11.48419869, 36.83936583],
       [11.12087087, 36.80837606],
       [10.97139243, 37.72236711],
       [11.58894858, 36.32214092],
       [10.70664152, 35.7661535 ],
       [11.07625934, 34.7797505 ],
       [12.95326254, 37.03427958],
       [10.5724666 , 36.86218335],
       [11.65803681, 37.42527868],
       [13.59251266, 36.83865679],
       [11.37176736, 35.26149812],
       [11.23341495, 37.21115258],
       [10.29035077, 36.92976156],
       [14.12178384, 38.40632901],
       [12.41554196, 37.67232248],
       [10.77024892, 34.64980005],
       [12.66439052, 36.36684325],
       [12.66780888, 37.48704925],
       [13.97018107, 36.67395274],
       [11.50925345, 36.59928909],
       [11.40577045, 36.37827101],
       [12.26371768, 38.86023443],
       [12.7107013 , 36.16646339],
       [13.47157767, 37.0716432 ],
       [11.18680872, 36.29889308],
       [11.24681261, 38.68258378],
       [12.35763811, 36.16604163],
       [11.76432596, 36.87502581],
       [11.7618838 , 38.1265198 ],
       [11.55030011, 35.76932963],
       [12.48266992, 35.53602453],
       [11.73136429, 36.07455114],
       [12.21407375, 37.19842782],
       [11.90375668, 36.8745436 ],
       [12.22248394, 36.35523488],
       [12.00591637, 36.53409567],
       [11.91374515, 36.0586476 ],
       [12.12540197, 38.18776368],
       [11.30555143, 37.13312676],
       [11.60899707, 37.68487728],
       [11.6930582 , 36.81293413],
       [11.20156988, 37.83544773],
       [12.62543264, 35.53914243],
       [13.35063168, 37.96597162],
       [13.56615961, 36.4179848 ],
       [11.69573629, 37.19026771],
       [11.49940906, 38.33257633],
       [12.39142299, 36.84008573],
       [12.41880832, 35.77101619]])
```

In [42]:

```
y = shop_data.iloc[:, 5].values
y
```

Out[42]:

```
array([587.951054 , 392.2049334, 487.5475049, 581.852344 , 599.406092 ,
       637.1024479, 521.5721748, 549.9041461, 570.200409 , 427.1993849,
       492.6060127, 522.3374046, 408.6403511, 573.4158673, 470.4527333,
       461.7807422, 457.8476959, 407.7045475, 452.3156755, 605.0610388,
       534.7057438, 419.9387748, 436.5156057, 519.3409891, 700.9170916,
       423.1799917, 619.8956399, 486.8389348, 529.5376653, 554.7220838,
       497.5866713, 447.6879065, 588.7126055, 491.0732237, 507.4418323,
       521.8835732, 347.7769266, 490.7386321, 478.1703341, 537.8461953,
       532.7517876, 501.8744303, 591.1971782, 547.2443434, 448.2298292,
       549.8605905, 593.915003 , 563.6728734, 479.7319491, 416.3583536,
       725.5848141, 442.6672517, 384.6265716, 451.4574469, 522.4041413,
       483.673308 , 520.8987945, 453.1695024, 496.6507081, 547.3651406,
       616.851523 , 507.212569 , 613.5993234, 483.1597208, 540.2634004,
       765.5184619, 553.6015347, 469.3108615, 408.6201878, 451.5756852,
```

```
444.9665517, 595.8228367, 418.1500811, 534.7771881, 578.2416051,
478.7193569, 444.2859075, 544.7798637, 488.7860611, 475.7590678,
489.812488 , 462.8976362, 596.4301726, 338.3198626, 533.5149353,
536.7718994, 487.379306 , 473.7289665, 547.1259317, 505.1133435,
449.0703194, 611.0000251, 515.8288149, 439.0747667, 514.0889577,
543.3401663, 521.1429518, 614.7153338, 507.3900618, 495.2994425,
518.064558 , 390.103273 , 420.7376732, 492.1050524, 410.0696111,
497.5136833, 494.5518611, 378.3309069, 570.4517259, 549.0082269,
459.2851235, 492.9450531, 424.7626355, 422.4267759, 642.1015787,
413.3717831, 479.2310929, 593.0772413, 506.5473071, 571.3074949,
576.3111774, 576.8025474, 514.2395207, 495.1759504, 514.3365583,
541.226584 , 516.8315567, 468.4457372, 548.2803202, 431.6177338,
552.9403455, 573.3062223, 452.627255 , 542.7115581, 407.8040306,
482.3535703, 529.2300901, 433.0487691, 476.1914133, 439.9978799,
448.9332932, 472.9922467, 463.923513 , 350.0582002, 460.0612774,
505.7711403, 463.4849954, 479.7319376, 424.1854943, 465.8893127,
426.775216 , 684.163431 , 555.8925954, 657.0199239, 595.8038189,
503.9783791, 586.1558702, 744.2218671, 512.8253581, 528.2238094,
468.9135013, 357.5914394, 536.4231045, 490.2066   , 550.0475806,
513.4505712, 497.81193  , 578.9862586, 506.5363931, 501.7492333,
421.9667942, 439.8912805, 666.1255917, 298.7620079, 465.1766233,
373.8857237, 532.7174857, 554.900783 , 537.7731625, 501.1002452,
517.1651356, 557.5292736, 493.719193 , 452.1226251, 577.273455 ,
485.9231305, 425.745092 , 537.2150527, 524.6379646, 478.8853913,
612.3852299, 476.7667242, 505.1196375, 545.9454921, 434.0216998,
424.675281 , 352.5501082, 662.9610878, 560.5601606, 467.5019004,
504.8704324, 590.5627196, 443.9656268, 392.4973992, 568.7175759,
712.3963268, 413.2959992, 562.0820454, 412.0129313, 468.6684656,
496.5540816, 548.5185293, 536.1308969, 558.4272572, 357.8637186,
529.0566632, 387.3570727, 528.9336186, 420.9161595, 496.9334463,
519.3729768, 591.4377356, 502.4097853, 604.3348401, 555.0683941,
256.6705823, 547.1109824, 461.9208769, 458.3769107, 436.2834981,
532.9352188, 512.5525344, 630.4227632, 463.7459811, 493.1802162,
501.2091727, 501.9282649, 376.3369008, 421.3266313, 538.7749335,
398.1634685, 571.4710341, 451.6286105, 490.6004425, 591.7810894,
409.0704721, 563.4460357, 647.6194557, 448.340425 , 518.7864831,
523.6339351, 393.857371 , 426.1545477, 503.3878873, 482.6024673,
524.7976276, 574.6548434, 574.7472197, 660.4251843, 375.3984554,
640.18774  , 514.0098178, 376.4968407, 484.5198091, 614.7296376,
567.4750105, 554.0030934, 399.9838716, 479.1728515, 585.9318443,
540.9957391, 628.0478039, 582.4919237, 640.7861664, 446.4186734,
570.6300981, 423.3083341, 616.660286 , 530.3624689, 442.3631174,
511.97986  , 560.4437922, 475.2634237, 374.2696745, 463.591418 ,
471.6028844, 626.0186727, 432.4720613, 356.6155679, 467.4278485,
503.2173931, 378.4735664, 584.2183135, 451.7278633, 557.634109 ,
432.7207178, 506.42386  , 510.1598173, 587.5747995, 282.4712457,
473.9498574, 489.9080531, 541.9722038, 266.0863409, 494.6871558,
689.7876042, 387.5347163, 441.8966315, 604.8413188, 302.1895478,
479.6148117, 506.1323424, 319.9288698, 528.309225 , 610.1280331,
584.105885 , 466.4211988, 404.8245289, 564.790969 , 596.516698 ,
368.6547849, 542.4124767, 478.2621264, 473.3604956, 559.199048 ,
447.1876443, 505.2300683, 557.2526867, 422.3687366, 445.0621855,
442.0644138, 533.0400602, 424.2028271, 498.6355985, 330.594446 ,
443.4418601, 478.6009159, 440.0027475, 357.7831107, 476.1392469,
501.1224915, 592.6884532, 486.0834255, 576.0252441, 442.7228916,
461.7909591, 488.3875258, 593.1564015, 392.810345 , 443.197221 ,
535.4807752, 533.3965538, 532.1274491, 558.9481124, 508.7719067,
403.7669021, 640.5840619, 461.6282784, 382.4161079, 561.8746577,
444.5761441, 401.0331352, 384.3260571, 527.7829958, 482.1449969,
594.2744834, 502.0925279, 407.6571788, 708.9351849, 531.9615505,
521.2407802, 447.3690272, 385.152338 , 430.5888826, 418.6027421,
478.9514048, 483.7965221, 538.9419745, 486.1637991, 385.0950071,
527.7837898, 547.1907494, 410.6029439, 583.977802 , 474.5323294,
414.9350607, 550.8133677, 458.7811317, 407.542168 , 581.3089329,
546.5566669, 503.1750852, 549.1315733, 482.8309859, 557.6082621,
484.8769649, 669.9871405, 547.7099886, 537.8252823, 408.2169018,
663.0748176, 506.3758668, 528.4193297, 632.1235881, 488.270298 ,
508.735741 , 411.1869636, 409.0945262, 467.8009244, 512.1658664,
608.2718166, 589.0264898, 444.0538266, 493.1812614, 532.7248055,
```

```
275.9184207, 511.0387861, 438.417742 , 475.7250679, 483.5431939,
663.8036933, 544.4092722, 630.1567282, 461.1122484, 491.9115051,
574.4156896, 530.7667187, 581.7987977, 556.2981412, 502.1327892,
556.1863689, 475.0716299, 486.9470538, 434.144202 , 304.1355916,
571.2160048, 583.0796357, 445.7498412, 392.9922559, 565.9943634,
499.1401524, 510.5394217, 308.5277466, 561.516532 , 423.4705332,
513.1531119, 529.1945189, 314.4385183, 478.584286 , 444.582165 ,
475.0154071, 436.7205559, 521.1953105, 478.1830597, 432.4811686,
438.3037078, 388.9405488, 534.7714849, 537.9157529, 407.8763782,
618.8459704, 502.7710746, 397.4205841, 392.2852442, 689.2356998,
543.1326263, 577.7360249, 436.5807403, 553.9946736, 427.3565308,
424.7287739, 541.049831 , 469.3831462, 444.5455497, 492.5568337,
535.3216101, 408.9583359, 487.5554581, 487.6462317, 402.1671222,
551.0230017, 497.3895578, 494.6386098, 479.2474168, 462.6565189,
515.5024797, 576.4776072, 357.8579836, 597.7398789, 327.3779526,
510.4013885, 510.5014785, 403.8195198, 627.6033187, 510.6617922,
573.8474377, 529.0490041, 551.6201455, 456.4695101, 497.7786422])
```

We are using `sklearn.model_selection.train_test_split` to split the data, using `test_size = 0.3` :

In [43]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(
 X, y, test_size=0.3, random_state=0
)
```

After that, we are splitting the training and testing datasets into different arrays for testing **Time on App** and **Time on Website**:

In [44]:
```
X_train_app = X_train[:, np.newaxis, 0]
X_test_app = X_test[:, np.newaxis, 0]

X_train_web = X_train[:, np.newaxis, 1]
X_test_web = X_test[:, np.newaxis, 1]
```

3. Regression

a. Training a Linear Regression Model

In this task, we are asked to train a linear regression model and report the coefficients of the regression model. Therefore, we are training two seperate linear regressions and comparing them.

App Linear Regression Model

Using `LinearRegression` , we are fitting a model to the training data for **Time on App**:

In [45]:
```
from sklearn.linear_model import LinearRegression
model_app = LinearRegression()
model_app.fit(X_train_app, y_train)
```

Out[45]:
```
LinearRegression()
```

We are going to use `.coef_` to report the coefficients, and `.intercept_` to report the intercept:

In [46]:
```
print('Intercept: \n', model_app.intercept_)
```

```
print('Coefficients: \n', model_app.coef_)
```

```
Intercept:
 29.061584912835713
Coefficients:
 [39.1629381]
```

Website Linear Regression Model

We are using `LinearRegression` again to fit a model to the training data for **Time on Website**:

In [47]:
```
model_web = LinearRegression()
model_web.fit(X_train_web, y_train)
```

Out[47]:
```
LinearRegression()
```

Using `.intercept_` and `.coef_`, we are able to report the Intercept and Coefficient of the regression model:

In [48]:
```
print('Intercept: \n', model_web.intercept_)
print('Coefficients: \n', model_web.coef_)
```

```
Intercept:
 560.3625431733054
Coefficients:
 [-1.58457927]
```

Comparison

Comparing the intercept and coefficients between the regression models of **Time on App** and **Time on Website**, we can see that:

- The intercept of App is **29.06** against the intercept of Web which is **560.36**.
  Therefore, the App Regression Model predicts that a customer will spend around **$29 yearly when they have spent 0 mins on the mobile app** whilst the Web Regression Model predicts that a customer will spend around **$560 yearly when they have spent 0 mins on the website**.
- The coefficient of App is **39.16** against the coefficient of Web which is **-1.58**.
  Therefore, the App Regression Model predicts that **every minute the customer spends on the mobile app will increase the amount they spend yearly by around $39** whilst the Web Regression Model predicts that **every minute will decrease yearly amount spent by $1.58**.

b.i. Prediction

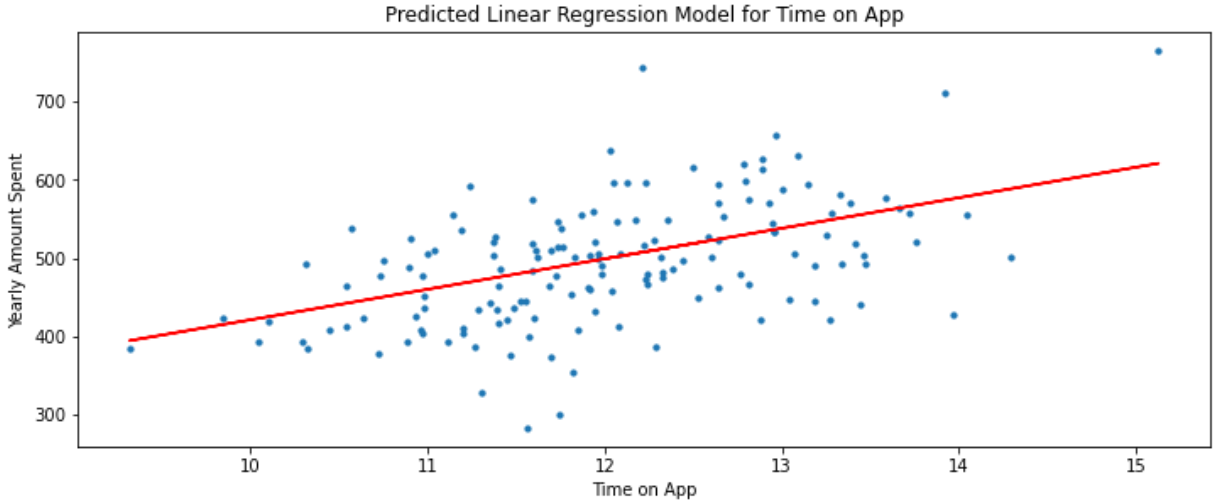
App Regression Model

Using `.predict()`, we are going to predict the results for the testing dataset for **Time on App**:

In [49]:
```
y_pred_app = model_app.predict(X_test_app)
```

Using `.scatter()` and `.plot`, we are creating a scatter plot of the testing data and plotting the predicted values from the regression model:

In [50]:
```
plt.figure(figsize=[12,4.5])
plt.scatter(X_test_app, y_test, s=10)
plt.plot(X_test_app, y_pred_app, color = 'r')
plt.xlabel('Time on App')
plt.ylabel('Yearly Amount Spent')
plt.title('Predicted Linear Regression Model for Time on App')
plt.show()
```



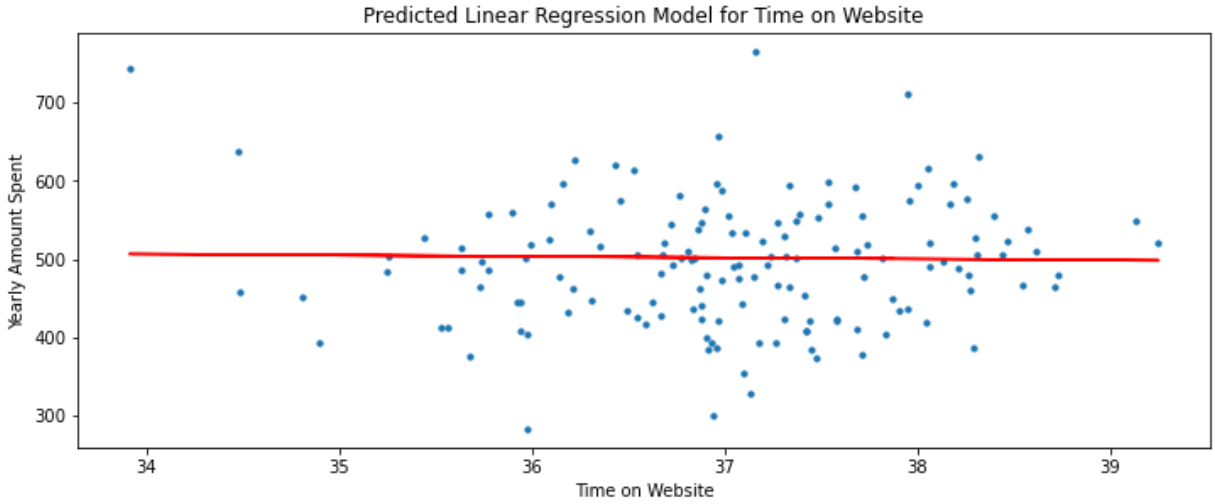
Web Regression Model

Using `.predict()`, we are going to predict the results for the testing dataset for **Time on Website**:

In [51]:
```
y_pred_web = model_web.predict(X_test_web)
```

Using `.scatter()` and `.plot`, we are creating a scatter plot of the testing data and plotting the predicted values from the regression model:

In [52]:
```
plt.figure(figsize=[12,4.5])
plt.scatter(X_test_web, y_test, s=10)
plt.plot(X_test_web, y_pred_web, color = 'r')
plt.xlabel('Time on Website')
plt.ylabel('Yearly Amount Spent')
plt.title('Predicted Linear Regression Model for Time on Website')
plt.show()
```



b.ii. Accuracy Metrics

Using `mean_squared_error` and `mean_absolute_error` from the `sklearn.metrics` package, we are going to report the MSE and MAE for both linear regression models.

In [53]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
```

App Linear Regression Model

In [55]:
```
mse_app = mean_squared_error(X_test_app, y_pred_app)
mae_app = mean_absolute_error(X_test_app, y_pred_app)
print('Mean Squared Error: \n', mse_app)
print('Mean Absolute Error: \n', mae_app)
```

```
Mean Squared Error:
 239359.95304028937
Mean Absolute Error:
 487.78085314255907
```

Web Linear Regression Model

In [56]:
```
mse_web = mean_squared_error(X_test_web, y_pred_web)
mae_web = mean_absolute_error(X_test_web, y_pred_web)
print('Mean Squared Error: \n', mse_web)
print('Mean Absolute Error: \n', mae_web)
```

```
Mean Squared Error:
 215908.44404061852
Mean Absolute Error:
 464.6528302674726
```

4. Conclusion

Based on the metrics obtained in 3.b.ii., the Web Linear Regression Model has a lower **mean squared error** and **mean absolute error** than the App Linear Regression Model.

The MSE and MAE are used to judge the accuracy of the regression model, with MSE taking the average of the squared errors between the predicted value and actual values whilst MAE averages the weighted differences equally.
The lower the value of MSE and MAE, the better the model's performance.

As the Web Linear Regression Model more accurately predicts the actual values of **Yearly Amount Spent**, the company should focus their efforts on their website.