Introduction

The National Longitudinal Survey of Youth 1997-2011 dataset is one of the most important databases available to social scientists working with US data.

It allows scientists to look at the determinants of earnings as well as educational attainment and has incredible relevance for government policy. It can also shed light on politically sensitive issues like how different educational attainment and salaries are for people of different ethnicity, sex, and other factors. When we have a better understanding how these variables affect education and earnings we can also formulate more suitable government policies.

Upgrade Plotly

In [1]:
```
%pip install --upgrade plotly
```
```
Requirement already satisfied: plotly in c:\users\manda\anaconda3\lib\site-packages (5.11.0)
Requirement already satisfied: tenacity>=6.2.0 in c:\users\manda\anaconda3\lib\site-packages (from plotly) (8.0.1)
Note: you may need to restart the kernel to use updated packages.
WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'ProtocolError('Connection abort
ed.', ConnectionResetError(10054, 'An existing connection was forcibly closed by the remote host', None, 10054, None))': /simple/plotly/
```

Import Statements

In [2]:
```
import pandas as pd
import numpy as np

import seaborn as sns
import plotly.express as px
import matplotlib.pyplot as plt

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
```

Notebook Presentation

In [3]:
```
pd.options.display.float_format = '{:,.2f}'.format
```

Load the Data

In [4]:
```
df_data = pd.read_csv('NLSY97_subset.csv')
```

Understand the Dataset

Have a look at the file entitled `NLSY97_Variable_Names_and_Descriptions.csv`.

```
:Key Variables:
  1. S           Years of schooling (highest grade completed as of 2011)
  2. EXP         Total out-of-school work experience (years) as of the 2011 interview.
  3. EARNINGS    Current hourly earnings in $ reported at the 2011 interview
```

Preliminary Data Exploration 🔎

Challenge

- What is the shape of `df_data`?
- How many rows and columns does it have?
- What are the column names?
- Are there any NaN values or duplicates?

In [5]:
```
print(f"df_data shape : {df_data.shape}")
print(f"df_data has {df_data.shape[0]} rows and {df_data.shape[1]} columns")
print(f"df_data column names: {df_data.columns}")
```
```
df_data shape : (2000, 96)
df_data has 2000 rows and 96 columns
df_data column names: Index(['ID', 'EARNINGS', 'S', 'EXP', 'FEMALE', 'MALE', 'BYEAR', 'AGE',
       'AGEMBTH', 'HHINC97', 'POVRAT97', 'HHBMBF', 'HHBMOF', 'HHOMBF',
       'HHBMONLY', 'HHBFONLY', 'HHOTHER', 'MSA97NO', 'MSA97NCC', 'MSA97CC',
       'MSA97NK', 'ETHBLACK', 'ETHHISP', 'ETHWHITE', 'EDUCPROF', 'EDUCPHD',
       'EDUCMAST', 'EDUCBA', 'EDUCAA', 'EDUCHSD', 'EDUCGED', 'EDUCDO',
       'PRMONM', 'PRMONF', 'PRMSTYUN', 'PRMSTYPE', 'PRMSTYAN', 'PRMSTYAE',
       'PRFSTYUN', 'PRFSTYPE', 'PRFSTYAN', 'PRFSTYAE', 'SINGLE', 'MARRIED',
       'COHABIT', 'OTHSING', 'FAITHN', 'FAITHP', 'FAITHC', 'FAITHJ', 'FAITHO',
       'FAITHM', 'ASVABAR', 'ASVABWK', 'ASVABPC', 'ASVABMK', 'ASVABNO',
       'ASVABCS', 'ASVABC', 'ASVABC4', 'VERBAL', 'ASVABMV', 'HEIGHT',
       'WEIGHT04', 'WEIGHT11', 'SF', 'SM', 'SFR', 'SMR', 'SIBLINGS', 'REG97NE',
       'REG97NC', 'REG97S', 'REG97W', 'RS97RURL', 'RS97URBN', 'RS97UNKN',
       'JOBS', 'HOURS', 'TENURE', 'CATGOV', 'CATPRI', 'CATNPO', 'CATMIS',
       'CATSE', 'COLLBARG', 'URBAN', 'REGNE', 'REGNC', 'REGW', 'REGS',
       'MSA11NO', 'MSA11NCC', 'MSA11CC', 'MSA11NK', 'MSA11NIC'],
      dtype='object')
```

In [6]:
```
df_data.isna().values.any()
```
Out[6]:
```
True
```

In [7]:
```
df_data.duplicated().values.any()
```
Out[7]:
```
True
```

Data Cleaning - Check for Missing Values and Duplicates

Find and remove any duplicate rows.

In [8]:
```
df_data.fillna(0, inplace=True)
```

In [9]:
```
df_data.drop_duplicates(inplace=True)
```

Descriptive Statistics

In [10]:
```
df_data.describe()
```

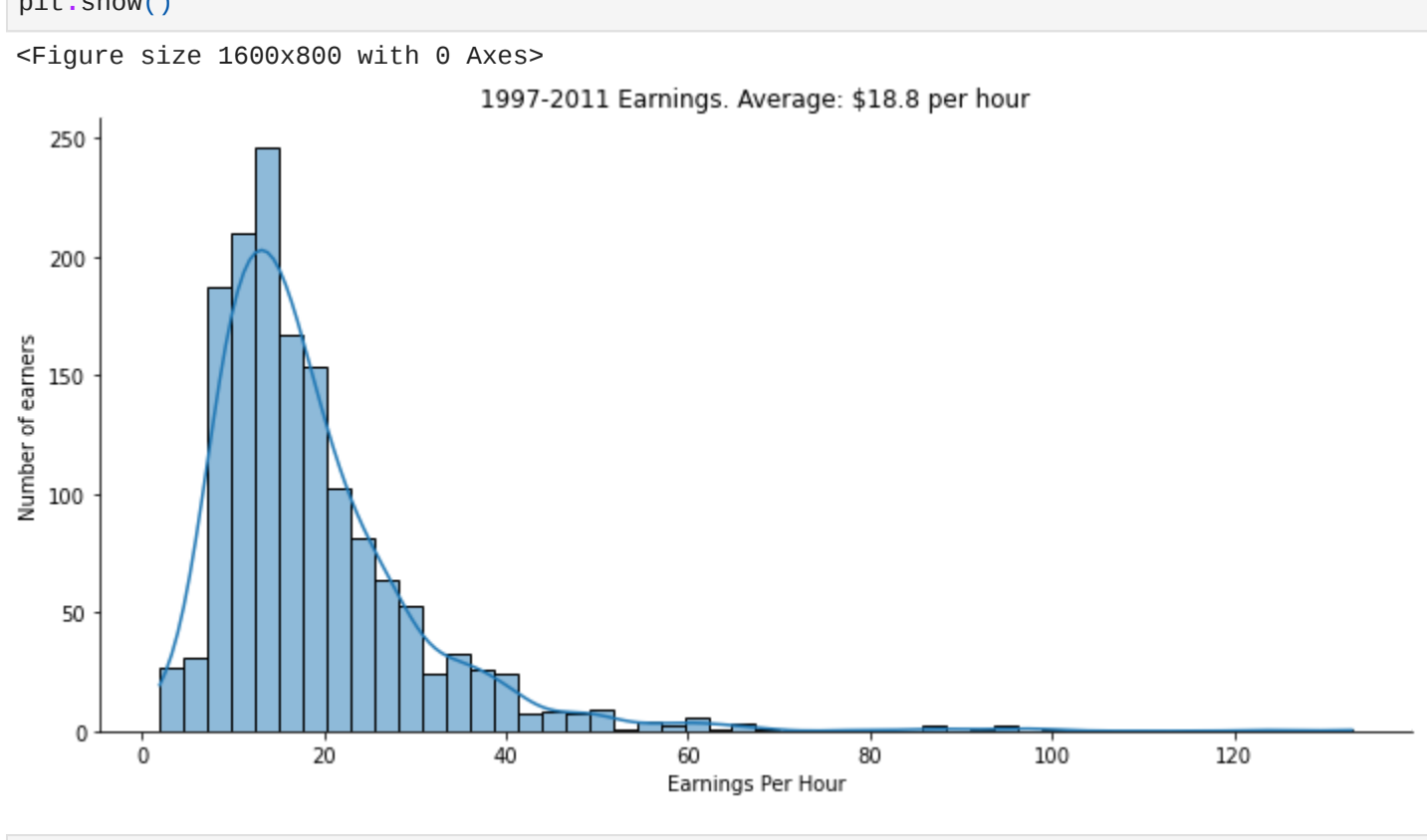
Out[10]:

	ID	EARNINGS	S	EXP	FEMALE	MALE	BYEAR	AGE	AGEMBTH	HHINC97	...	URBAN	REGNE	REGNC	REGW	REGS	MSA11NO	MSA11NCC
count	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	...	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00	1,487.00
mean	3,547.13	18.81	14.56	6.70	0.49	0.51	1,982.07	28.93	25.72	47,252.43	...	0.78	0.16	0.27	0.34	0.23	0.05	0.54
std	2,009.84	12.00	2.77	2.86	0.50	0.50	1.38	1.38	6.38	45,629.92	...	0.43	0.36	0.44	0.47	0.42	0.21	0.50
min	1.00	2.00	6.00	0.00	0.00	0.00	1,980.00	27.00	0.00	0.00	...	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	1,888.00	11.41	12.00	4.65	0.00	0.00	1,981.00	28.00	22.00	13,740.50	...	1.00	0.00	0.00	0.00	0.00	0.00	0.00
50%	3,474.00	15.75	15.00	6.63	0.00	1.00	1,982.00	29.00	26.00	41,840.00	...	1.00	0.00	0.00	0.00	0.00	0.00	1.00
75%	5,160.50	22.60	16.00	8.71	1.00	1.00	1,983.00	30.00	30.00	66,400.00	...	1.00	0.00	1.00	1.00	0.00	0.00	1.00
max	8,980.00	132.89	20.00	14.73	1.00	1.00	1,984.00	31.00	45.00	246,474.00	...	2.00	1.00	1.00	1.00	1.00	1.00	1.00

8 rows × 96 columns

Visualise the Features

In [11]:
```
plt.figure(figsize=(8, 4),dpi=200)
sns.displot(df_data["EARNINGS"], bins=50, aspect=2, kde=True)
plt.title(f'1997-2011 Earnings. Average: ${(df_data.EARNINGS.mean()):.3} per hour')
plt.xlabel("Earnings Per Hour")
plt.ylabel("Number of earners")
plt.show()
```
```
<Figure size 1600x800 with 0 Axes>
```



In [12]:
```
df_data["EDUCBA"]
```
Out[12]:
```
0       0
1       1
2       0
3       0
4       0
       ..
1984    0
1989    0
1991    1
1994    0
1995    0
Name: EDUCBA, Length: 1487, dtype: int64
```

Split Training & Test Dataset

We *can't* use all the entries in our dataset to train our model. Keep 20% of the data for later as a testing dataset (out-of-sample data).

In [13]:
```
y = df_data["EARNINGS"]
to_drop = ["EARNINGS"]
X = df_data.drop(to_drop, axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=10)
```

In [14]:
```
train_pct = 100 * len(X_train) / len(X)
print(f"Training data is : {train_pct:.3}% of the total data")
```
```
Training data is : 80.0% of the total data
```

In [15]:
```
test_pct = 100 * X_test.shape[0] / X.shape[0]
print(f"Test data makes up the remaining {test_pct:.3}%.")
```
```
Test data makes up the remaining 20.0%.
```

Simple Linear Regression

Only use the years of schooling to predict earnings. Use sklearn to run the regression on the training dataset. How high is the r-squared for the regression on the training data?

In [16]:
```
X_axis=pd.DataFrame(X_train["S"])
X_axis.head()
```

Out[16]:

	S
179	14
142	16
1626	10
707	12
1288	18

In [17]:
```
lm = LinearRegression()
```

In [18]:
```
model = lm.fit(X_axis, y_train)
```

In [19]:
```
print(f"r-squared the years of schooling to predict earnings: {model.score(X_axis, y_train):.2}")
```
```
r-squared the years of schooling to predict earnings: 0.081
```

In [20]:
```
print("Conclusion: Using Years of Schooling is not enough variable to predict earnings.")
```
```
Conclusion: Using Years of Schooling is not enough variable to predict earnings.
```

Evaluate the Coefficients of the Model

Here we do a sense check on our regression coefficients. The first thing to look for is if the coefficients have the expected sign (positive or negative).

Interpret the regression. How many extra dollars can one expect to earn for an additional year of schooling?

In [21]:
```
print(f"model predict : {model.predict(X_axis)}")
theta_1 = model.coef_
print(f"model coef: {theta_1}")
print(f"model incercept: {model.intercept_}")
```
```
model predict : [18.09739916 20.53575512 13.22068723 ... 18.09739916 20.53575512
 20.53575512]
model coef: [1.21917798]
model incercept: 1.0289074107474185
```

In [22]:
```
coef_regr = pd.DataFrame(data=theta_1, index=X_axis.columns, columns=["Coeffisien"])
coef_regr
```

Out[22]:

	Coeffisien
S	1.22

In [23]:
```
year_school = coef_regr.loc["S"].values[0]
print(f"extra dollar to expect for an additional year of schooling : ${year_school:.3} per hour.")
```
```
extra dollar to expect for an additional year of schooling : $1.22 per hour.
```

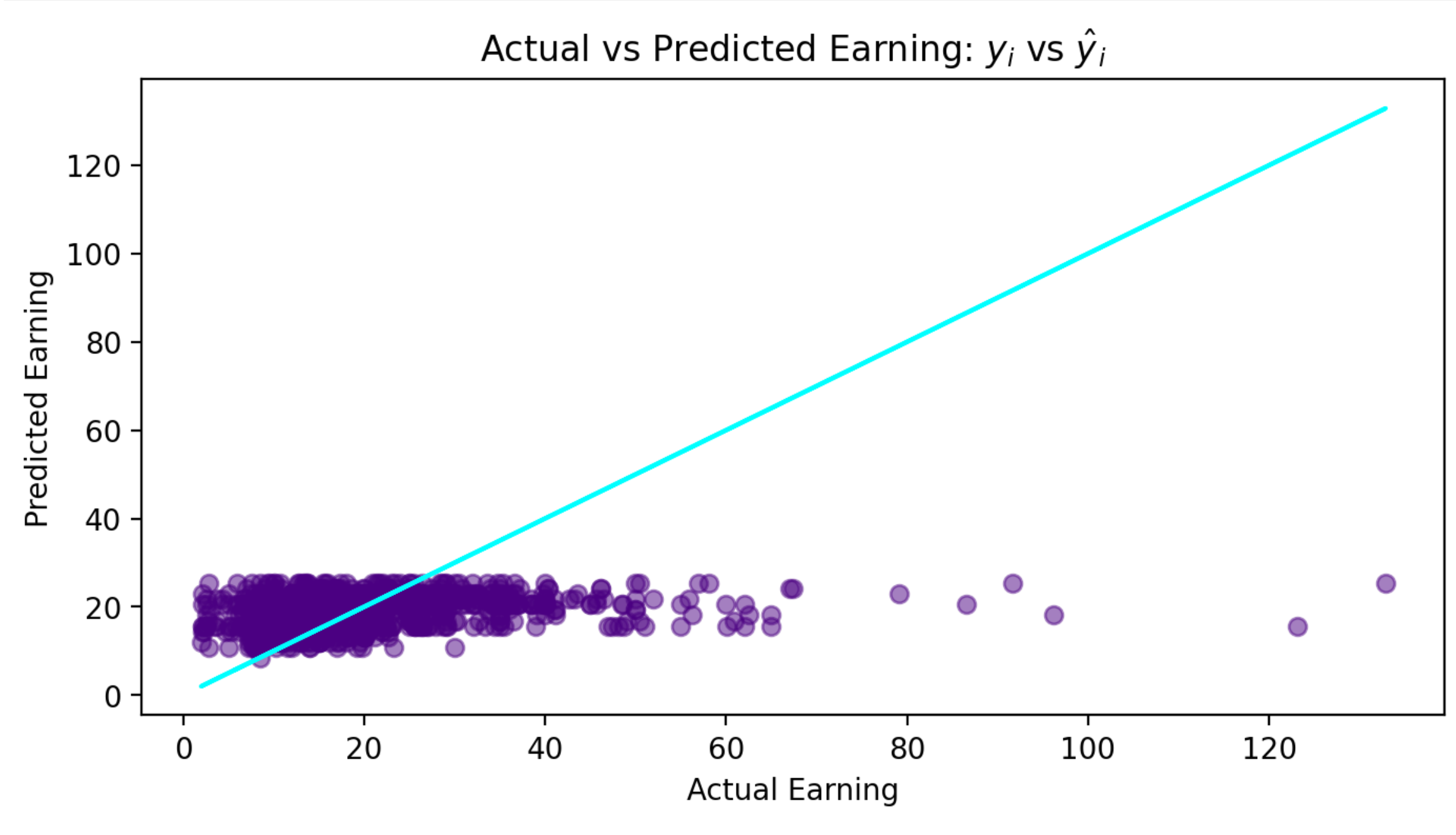
Analyse the Estimated Values & Regression Residuals

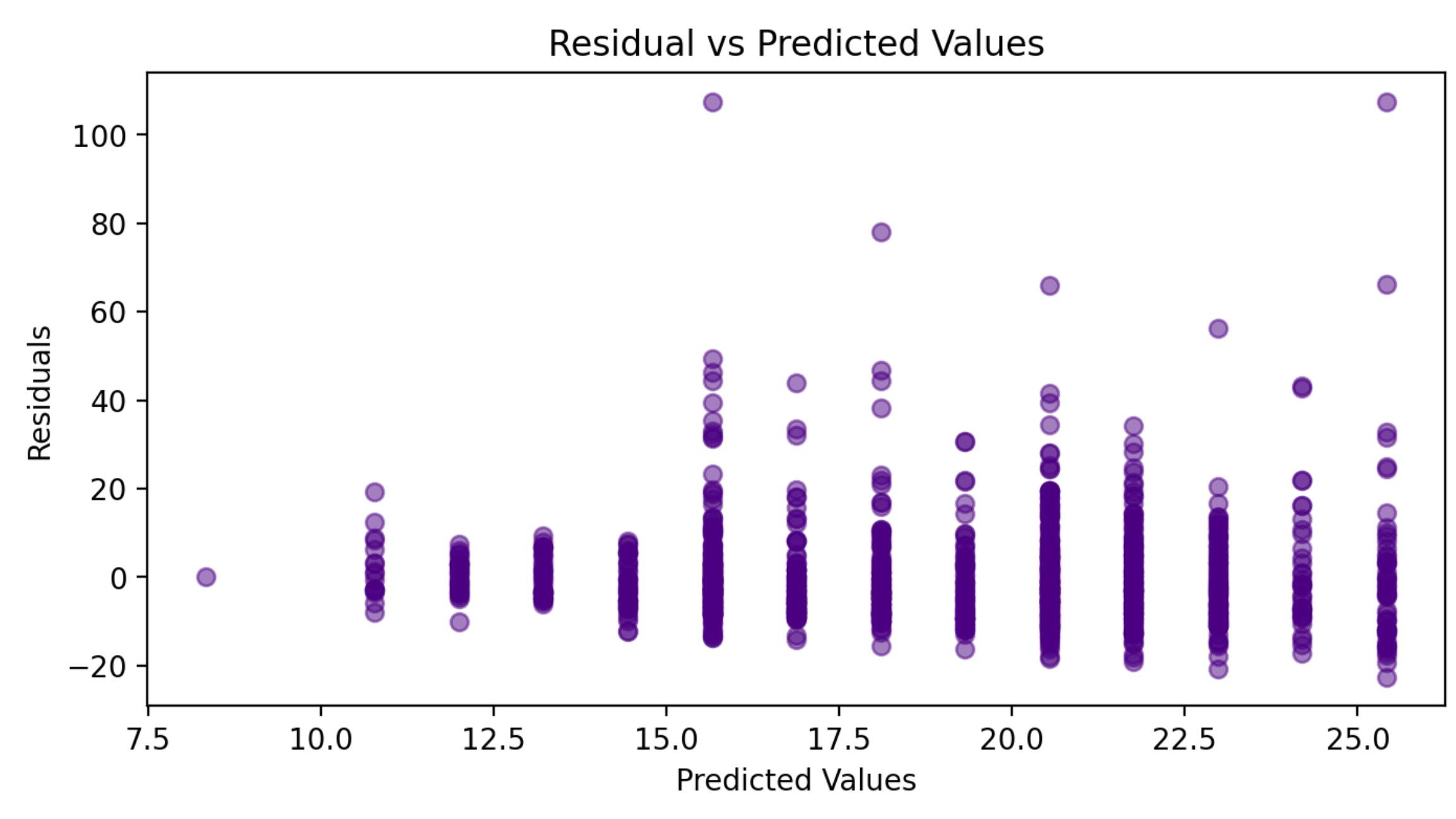
How good our regression is also depends on the residuals - the difference between the model's predictions ($\hat{y}_i$) and the true values (y_i) inside y_train. Do you see any patterns in the distribution of the residuals?

In [24]:
```
predicted_values = model.predict(X_axis)
residuals = (y_train - predicted_values)
print(f"predicted values \n {predicted_values}")
print(f"residuals \n {residuals}")
```
```
predicted values
 [18.09739916 20.53575512 13.22068723 ... 18.09739916 20.53575512
 20.53575512]
residuals
 179    -12.34
142     -5.16
1626     6.78
707      2.12
1288    -6.97
        ...
1762    23.92
1660     7.50
527      7.90
1263    19.46
1541     5.90
Name: EARNINGS, Length: 1189, dtype: float64
```

In [25]:
```
plt.figure(figsize=(8,4), dpi=200)
plt.scatter(x=y_train, y=predicted_values, c="indigo", alpha=0.5)
plt.plot(y_train, y_train, color="cyan")
plt.title("Actual vs Predicted Earning: $y _i$ vs $\hat y_i$")
plt.xlabel("Actual Earning")
plt.ylabel("Predicted Earning")
plt.show()


plt.figure(figsize=(8,4), dpi=200)
plt.scatter(x=predicted_values, y=residuals, c="indigo", alpha=0.5)
plt.title("Residual vs Predicted Values")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.show()
```





Multivariable Regression

Now use both years of schooling and the years work experience to predict earnings. How high is the r-squared for the regression on the training data?

In [26]:
```
new_X = df_data[["S", "EXP"]]
```

In [27]:
```
new_Y = df_data["EARNINGS"]
```

In [28]:
```
X_train, X_test, EXS_y_train, EXS_y_test = train_test_split(new_X,
                                                    new_Y,
                                                    test_size=0.2,
                                                    random_state=10)

regr = LinearRegression()
regr.fit(X_train, EXS_y_train)
rsquared = regr.score(X_train, EXS_y_train)

EXS_predictions = regr.predict(X_train)
residuals = (EXS_y_train - EXS_predictions)

print(f'Training data r-squared: {rsquared:.3}')
```
```
Training data r-squared: 0.117
```

Evaluate the Coefficients of the Model

In [29]:
```
new_coef = regr.coef_
new_coef
```
Out[29]:
```
array([1.79422175, 0.97900689])
```

In [30]:
```
df_coef = pd.DataFrame(new_coef, index=X_train.columns, columns=['coef'])
df_coef
```

Out[30]:

	coef
S	1.79
EXP	0.98

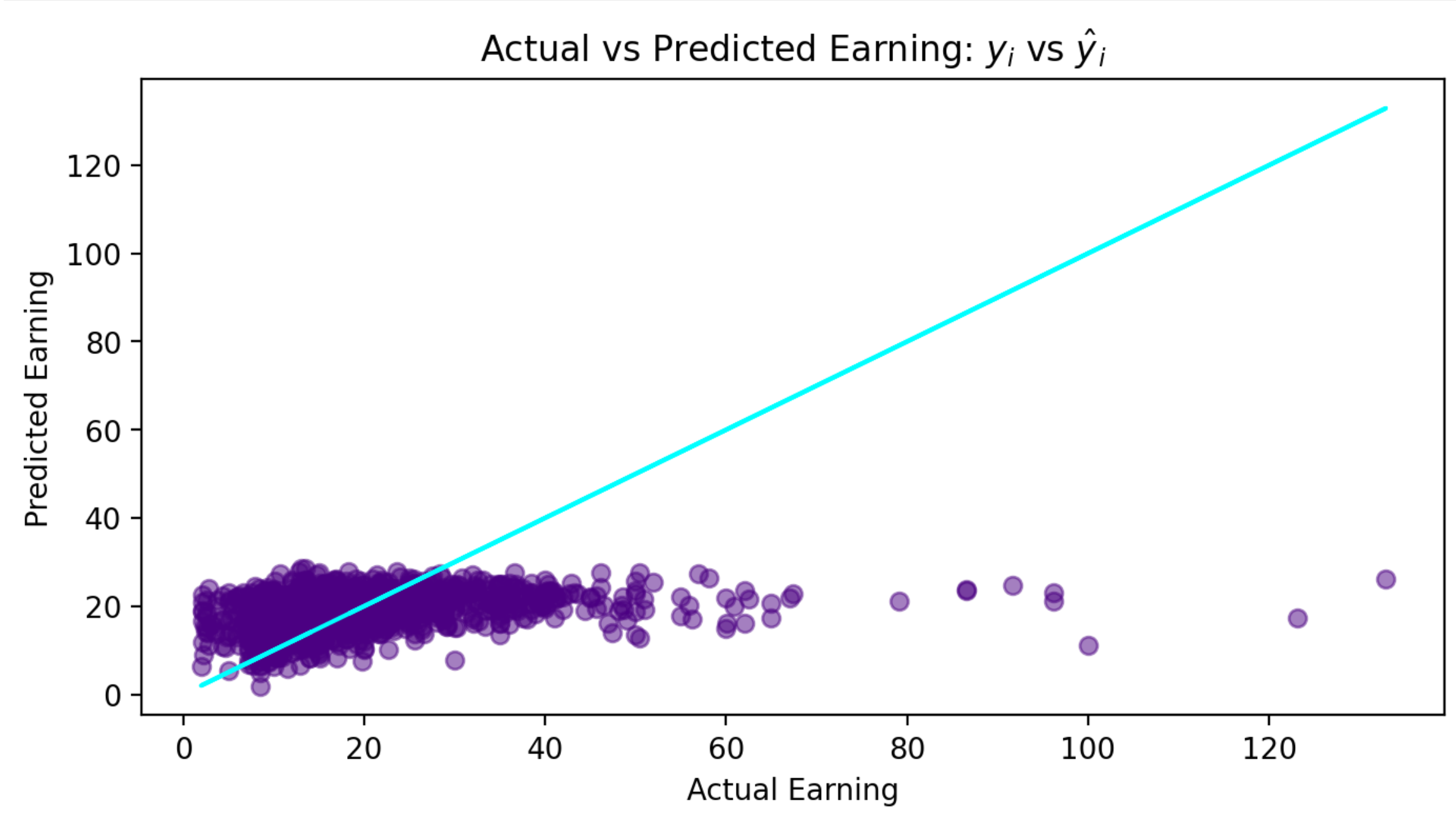
In [31]:
```
print("The coefficients interpret: having additional year of school and more work experience results in higher earning, because S and EXP has a coef
```
```
The coefficients interpret: having additional year of school and more work experience results in higher earning, because S and EXP has a coefficien
t greater than zero.
```

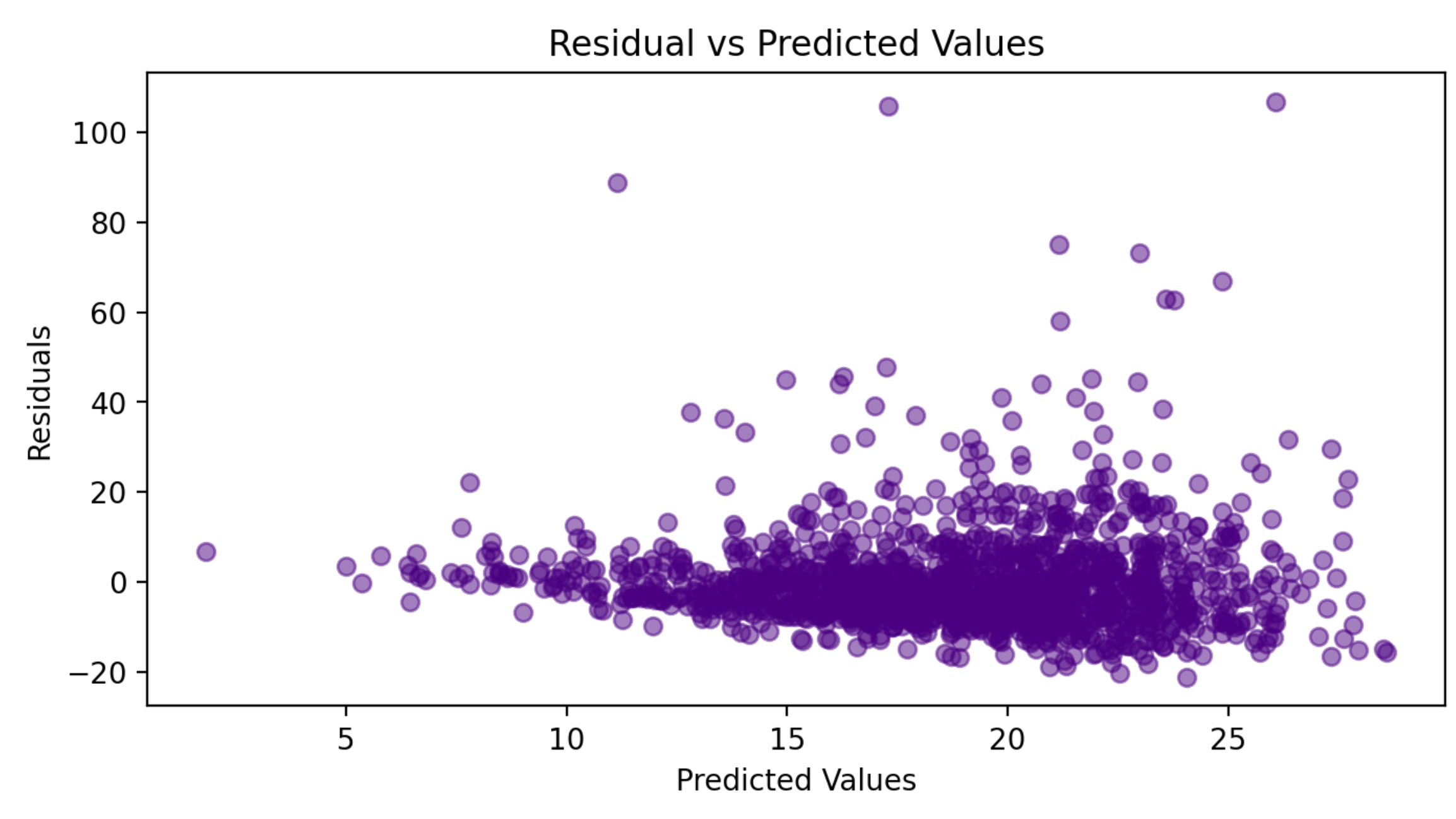
Analyse the Estimated Values & Regression Residuals

In [32]:
```
EXP_predicted_values = regr.predict(new_X)
EXP_residuals = (new_Y - EXP_predicted_values)
```

In [33]:
```
plt.figure(figsize=(8,4), dpi=200)
plt.scatter(x=new_Y, y=EXP_predicted_values, c="indigo", alpha=0.5)
plt.plot(new_Y, new_Y, color="cyan")
plt.title("Actual vs Predicted Earning: $y _i$ vs $\hat y_i$")
plt.xlabel("Actual Earning")
plt.ylabel("Predicted Earning")
plt.show()


plt.figure(figsize=(8,4), dpi=200)
plt.scatter(x=EXP_predicted_values, y=EXP_residuals, c="indigo", alpha=0.5)
plt.title("Residual vs Predicted Values")
plt.xlabel("Predicted Values")
plt.ylabel("Residuals")
plt.show()
```





Use Your Model to Make a Prediction

How much can someone with a bachelors degree (12 + 4) years of schooling and 5 years work experience expect to earn in 2011?

In [34]:
```
school = 16
experience = 5

# Starting Point: Average Values in the Dataset
features = df_data[['S', 'EXP']]
average_vals = features.mean().values
someone = pd.DataFrame(data=average_vals.reshape(1, len(features.columns)),
                              columns=features.columns)
someone
```

Out[34]:

	S	EXP
0	14.56	6.70

In [35]:
```
#SOLUTION

someone['S'] = school
someone['EXP'] = experience
someone
```

Out[35]:

	S	EXP
0	16	5

In [36]:
```
# making prediction
estimate = regr.predict(someone)[0]
print(f"the earning estimate is : ${estimate:.3} per hour.")
```
```
the earning estimate is : $19.7 per hour.
```

Experiment and Investigate Further

Which other features could you consider adding to further improve the regression to better predict earnings?

In [37]:
```
df_data
```

Out[37]:

	ID	EARNINGS	S	EXP	FEMALE	MALE	BYEAR	AGE	AGEMBTH	HHINC97	...	URBAN	REGNE	REGNC	REGW	REGS	MSA11NO	MSA11NCC	MSA11CC	MSA11NK	M
0	4275	18.50	12	9.71	0	1	1984	27	24.00	64,000.00	...	1	0	0	1	0	0	0	1	0	
1	4328	19.23	17	5.71	0	1	1982	29	32.00	6,000.00	...	2	0	0	1	0	0	1	0	0	
2	8763	39.05	14	9.94	0	1	1981	30	23.00	88,252.00	...	1	0	0	0	1	0	0	1	0	
3	8879	16.80	18	1.54	0	1	1983	28	30.00	0.00	...	1	0	1	0	0	0	1	0	0	
4	1994	36.06	15	2.94	0	1	1984	27	23.00	44,188.00	...	1	0	0	0	1	0	0	1	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
1984	2400	9.00	12	10.83	1	0	1982	29	29.00	73,100.00	...	0	0	0	1	0	0	0	1	0	
1989	3018	8.00	12	9.37	1	0	1982	29	23.00	66,300.00	...	1	0	1	0	0	0	1	0	0	
1991	4550	8.57	17	6.29	1	0	1984	27	25.00	76,300.00	...	0	0	0	0	1	0	1	0	0	
1994	3779	9.33	12	9.12	1	0	1984	27	22.00	0.00	...	1	0	0	1	0	0	1	0	0	
1995	2456	14.00	8	7.87	1	0	1982	29	19.00	6,000.00	...	1	1	0	0	0	0	1	0	0	

1487 rows × 96 columns

In [38]:
```
new_features = df_data[['EXP', 'S', 'MALE']]
new_y = df_data['EARNINGS']
new_features
```

Out[38]:

	EXP	S	MALE
0	9.71	12	1
1	5.71	17	1
2	9.94	14	1
3	1.54	18	1
4	2.94	15	1
...	...	...	...
1984	10.83	12	0
1989	9.37	12	0
1991	6.29	17	0
1994	9.12	12	0
1995	7.87	8	0

1487 rows × 3 columns

In [39]:
```
LX_train, X_test, L_y_train, L_y_test = train_test_split(new_features,
                                                    new_y,
                                                    test_size=0.2,
                                                    random_state=10)
```

In [40]:
```
Lregr = LinearRegression()
Lregr.fit(LX_train, L_y_train)
Lrsquared = Lregr.score(LX_train, L_y_train)
```

In [41]:
```
L_predictions = Lregr.predict(LX_train)
residuals = (L_y_train - L_predictions)

print(f'Training data r-squared: {Lrsquared:.2}')
```
```
Training data r-squared: 0.15
```

In [42]:
```
L_coef = Lregr.coef_
L_coef
```
Out[42]:
```
array([0.99649898, 1.92744945, 4.2323065 ])
```

In [43]:
```
Ldf_coef = pd.DataFrame(L_coef, index=LX_train.columns, columns=['coef'])
Ldf_coef.sort_values("coef", ascending=False)
```

Out[43]:

	coef
MALE	4.23
S	1.93
EXP	1.00

In [44]:
```
print("The coefficients interpret: being a male worker, and having additional year of school and more work experience results in higher earning, bec
```
```
The coefficients interpret: being a male worker, and having additional year of school and more work experience results in higher earning, because M
ALE, S and EXP has a coefficient greater than zero.
```